NATO RTG SAS-045
on
Computer Based Decision Support Tool for Helicopter Mission Planning in Disaster Relief and Military Operations

Annex A – TECHNICAL REPORT 1: OVERVIEW OF DECISION SUPPORT TOOL TECHNOLOGIES

Mickaël Fontaine
DSP/CAD, 16bis avenue Prieur de la cote d'Or, 94114 Arcueil, France
Tel: [33] (1) 42 31 90 58; Fax: [33] (1) 42 31 91 75

E-mail: mickael.fontaine@dga.defense.gouv.fr

A.1 INTRODUCTION

This document is prepared to fulfill the requirements of the Programme of Work (POW) of NATO SAS-045 RTG on "Computer Based Decision Support Tool for Helicopter Mission Planning in Disaster Relief and Military Operations".

The POW dictates to provide an a-priori analysis of modeling, computer (software engineering) and data collection technologies that are anticipated to support the development of NATO SAS-045 project. As it is stated in the Terms of Reference (TOR) of the afore-mentioned project, the main goal of the research is to provide the basis for developing a generic and flexible decision support tool for effective management of helicopter missions by conducting the problem analysis, investigating the concept of solutions and determining relevant technical requirements.

It is foreseen that different models, technologies and solution approaches will be utilized concurrently, and integration and interfacing will pose itself as an important technical issue within the scope of maintaining interoperability and standardization in NATO practices.

Thus, the POW states that during the Analysis Phase of the project technology surveys should be carried out on modeling, computer and software engineering and data collection technologies; geographical information systems, digital maps, mission data compilation systems; model, data, and knowledge management repositories in NATO nations. Then, the technology mapping and capability matrix can be developed using the identified current needs and capability gaps.

In this document, it is intended to provide a broad overview that will guide any potential research dealing with the design and development of a generic decision support tool within a similar NATO context, not limited to helicopter operations.

The aim of this document is not to cover in detail the optimization methods but rather to give a general idea of techniques and products. In this report, we provide information on Operations research techniques Simulation Data Analysis & Mining and Artificial intelligence in order to determine the design requirements for the helicopter mission planning decision support system. For more detailed operations research techniques' information see the references.

A.2 OPTIMIZATION OVERVIEW

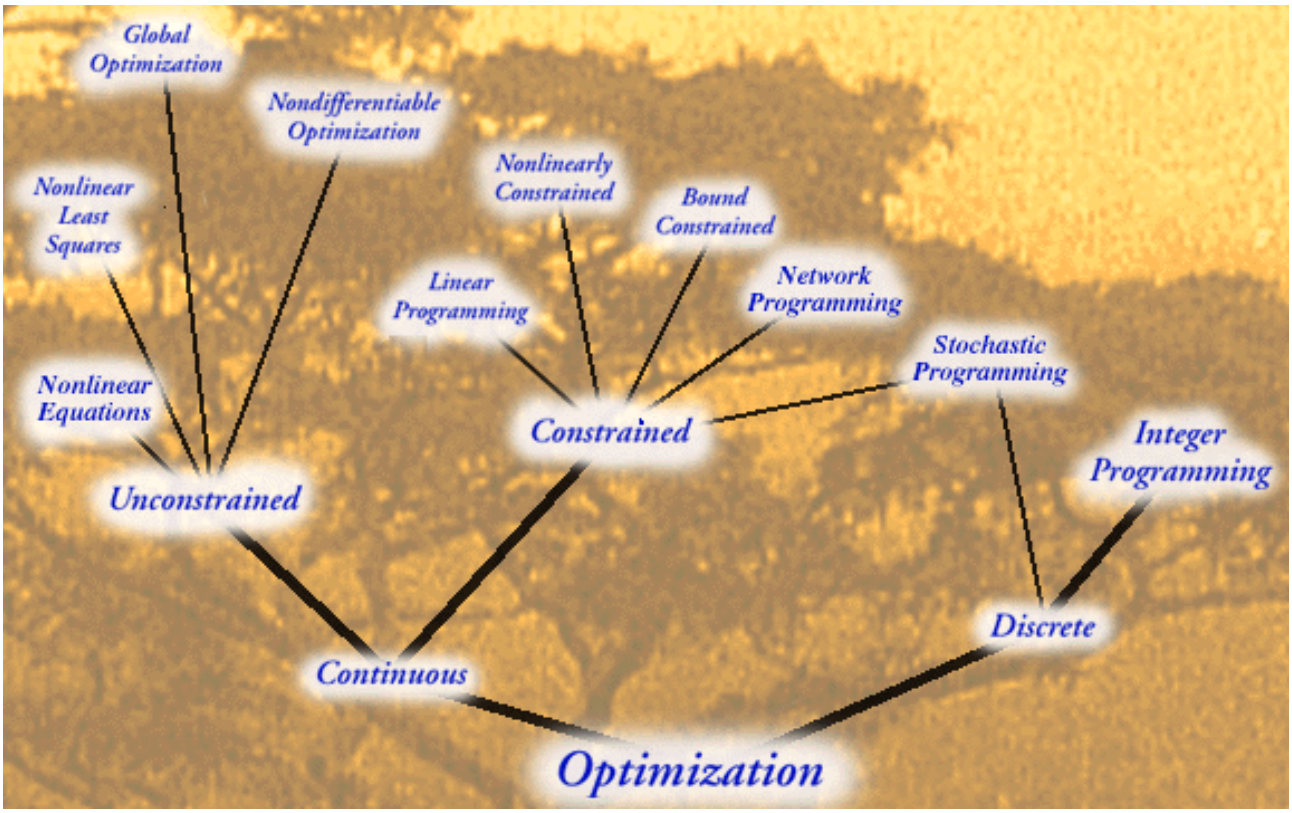

Figure A.1: A View of Different Subfields of Optimization.

This part introduces the different subfields of **optimization (or mathematical programming)** and includes outlines of the major algorithms in each area, with pointers to software packages where appropriate.

A.2.1 Introduction

Optimization problems are made up of three basic ingredients:

- An **objective function** we want to minimize or maximize. For instance, in a manufacturing process, we might want to maximize the profit or minimize the cost. In fitting experimental data to a user-defined model, we might minimize the total deviation of observed data from predictions based on the model. In designing an automobile panel, we might want to maximize the strength.
- A set of **unknowns** or **variables** affect the value of the objective function. In the manufacturing problem, the variables might include the amounts of different resources used or the time spent on each activity. In fitting-the-data problem, the unknowns are the parameters that define the model. In the panel design problem, the variables used define the shape and dimensions of the panel.
- A set of **constraints** that allow the unknowns to take on certain values but exclude others. For the manufacturing problem, it does not make sense to spend a negative amount of time on any activity, so we constrain all the "time" variables to be non-negative. In the panel design problem, we would probably want to limit the weight of the product and to constrain its shape.

The optimization problem is then:

Find values of the variables that minimize or maximize the objective function while satisfying the constraints.

Are All these ingredients necessary?

Objective Function

Almost all optimization problems have a single objective function. (When they don't they can often be reformulated so that they do!) The two interesting exceptions are:

- No objective function. In some cases (for example, design of integrated circuit layouts), the goal is to find a set of variables that satisfies the constraints of the model. The user does not particularly want to optimize anything so there is no reason to define an objective function. This type of problems is usually called a feasibility problem.
- Multiple objective functions. Often, the user would actually like to optimize a number of different objectives at once. For instance, in the panel design problem, it would be nice to minimize weight and maximize strength simultaneously. Usually, the different objectives are not compatible; the variables that optimize one objective may be far from optimal for the others. In practice, problems with multiple objectives are reformulated as single-objective problems by either forming a weighted combination of the different objectives or else replacing some of the objectives by constraints.

Variables

These are essential. If there are no variables, we cannot define the objective function and the problem constraints.

Constraints

Constraints are not essential. In fact, the field of unconstrained optimization is a large and important one for which a lot of algorithms and software are available. It's been argued that almost all problems really do have constraints. For example, any variable denoting the "number of objects" in a system can only be useful if it is less than the number of elementary particles in the known universe! In practice though, answers that make good sense in terms of the underlying physical or economic problem can often be obtained without putting constraints on the variables.

A.2.1.1 Mathematical Programming (Optimization) Theory

Many decision problems can be modeled using mathematical programs, which seek to maximize or minimize some objective which is a function of the decisions. The possible decisions are constrained by limits in resources, minimum requirements, etc. Decisions are represented by variables, which may be, for example, non-negative or integer. Objectives and constraints are functions of the variables, and problem data. Examples of problem data include unit costs, production rates, sales, or capacities.

Suppose the decisions are represented by the variables (x1, x2, ...xn). For example, x(i) can represent production of the i th of n products. The general form of a mathematical program is

Minimize $f(x_1, x_2, x_3, ..., x_n)$
subject to $g1(x_1, x_2, x_3, ..., x_n) <= 0$
…
$gm(x_1, x_2, x_3, ..., x_n) <= 0$
$x_1, x_2, x_3, ..., x_n$ in X

where X is a set that be, e.g. all non-negative real numbers. The constraints can be quite general, but linear constraints are sufficient in many cases to capture the essence of the model.

A.2.2 Linear Programming

A.2.2.1 Linear Programming Presentation

A Linear Program (LP) is a problem that can be expressed as follows (the so-called Standard Form):

minimize cx
subject to Ax = b
x > = 0

where x is the vector of variables to be solved for, A is a matrix of known coefficients, and c and b are vectors of known coefficients. The expression "cx" is called the objective function, and the equations "Ax = b" are called the constraints. All these entities must have consistent dimensions, of course, and you can add "transpose" symbols to taste. The matrix A is generally not square, hence you don't solve an LP by just inverting A. Usually A has more columns than rows, and Ax = b is therefore quite likely to be under-determined, leaving great latitude in the choice of x with which to minimize cx.

The word "Programming" is used here in the sense of "planning"; the necessary relationship to computer programming was incidental to the choice of name.

A.2.3 Non-Linear Programming

At least one of the functions (objective or constraint) is not affine (Usually, it is the rule of the function that classifies it as non-linear. In particular, a linear integer program is not generally considered an NLP.)

A.2.4 Integer Programming

Many times decision variables only have meaning when they are integers. Integer Linear Programs are similar to Linear Programs except that decision variables can only take on integer values. They are modeled like linear programs with additional constraints that variables assume integer values.

A.2.5 Mathematical Programming Products

A.2.5.1 CPLEX

The main module of CPLEX is the Linear Optimizer Base System. It is a linear programming environment which includes fast optimization algorithms as well as a full set of utilities to support solving linear programming problems. The Callable Library is an optimization library system created specifically for those with development or specialized application requirements. The Mixed Integer Solver is a high-performance optimizer for problems with integer variables while the Barrier/QP Solver is an optimizer for solving linear and quadratic programming problems available as an option with CPLEX optimization software. There are also Parallel Solvers taking full advantage of selected parallel computer architectures to deliver high solution performance.

OS: IBM-compatible PCs, Macintosh, various UNIX systems

Vendor: ILOG CPLEX Division http://www.cplex.com

A.2.5.2 XPRESS-MP

XPRESS-MP is a complete Linear and Integer Programming modelling and optimization system built around a state-of-the-art model builder and Simplex and Interior Point optimizers. XPRESS-MP is designed for the professional programmer. It helps you build, test and debug MP models on a personal computer, workstation, mini or mainframe faster than ever before. Because XPRESS-MP looks identical on a variety of hardware platforms, you can develop your application where it is most convenient for you and base your production system where it is most convenient for end users.

OS: Windows 95, Windows 98, Windows NT, UNIX, Linux, Solaris, AIX, etc.

Vendor: Dash Associates Ltd. http://www.dash.co.uk

Demo: http://www.dash.co.uk/evaluate.html

A.2.5.3 LINDO

LINDO is one of the most popular Linear Programming engines. It solves linear, integer, and quadratic optimization problems providing speedy problem entry, solution, and analysis. LINDO's straightforward, simple style makes it easy for beginners to learn. And, for developers, there's the flexibility of being able to link in your own routines, add custom commands and use the LINDO engine to run your own vehicle.

OS: Available for PC and (in Hyper and larger sizes) the following workstations: IBM, HP 9000, SunSPARC, Linux, Silicon Graphics, and DEC Alpha.

Vendor: LINDO Systems http://www.lindo.com

Demo: http://www.lindo.com/cgi-bin/frameset.cgi?leftdwnld.html;downloadf.html

A.2.5.4 OSL

OSL is high quality optimization software implementing fast, robust, state-of-the-art algorithms. The Optimization Solutions products provide stand alone solver capabilities for analyzing linear programming (including network programming) problems, quadratic programming problems, mixed integer programming problems, and stochastic programming problems.

These Solutions incorporate modules from the Optimization Library, and provide most of the functionality of the complete library without requiring the user to develop code. For linear programming and quadratic programming problems, both simplex solvers and interior point solvers are included.

The size of problems that the optimization solutions programs can handle is limited only by available memory. The Optimization Library includes about 70 user callable functions for manipulating models and analyzing the resulting mathematical problems. Besides the solvers, there are modules that analyze, simplify, and transform problems prior to solution, and others that assist with analyzing results.

These post solution analysis capabilities include: infeasibility, sensitivity, and parametrics. The modules of the Optimization Library include many program options, control variables, and user exit function calls that provide great flexibility in creating specialized applications. The Optimization Solutions modules provide access to the program options and the control variables.

OS: RISC System/6000 running AIX , PCs running Windows (NT or 95), and UNIX-based workstations from Hewlett Packard, Silicon Graphics, and Sun.

Vendor: IBM http://www6.software.ibm.com/es/oslv2/features/welcome.htm

Demo: http://www6.software.ibm.com/es/oslv2/features/download.htm

A.2.5.5 LP Solve (Non-Commercial Software)

The linear programming problem can be formulated as: Solve Ax – V_1 , with V_2 x maximal. A is a matrix, x a vector of (non-negative) variables, V_1 a vector called the right hand side, and V_2 a vector specifying the objective function.

Any number of the variables may be specified to be of type integer. This program solves problems of this kind. It is slightly more general than the above problem, in that every row of A (specifying one constraint) can have its own (in)equality, Ÿ, – or =. The result specifies values for all variables. LP Solve uses a ``Simplex'' algorithm and sparse matrix methods, for pure LP problems. If one or more of the variables is declared integer, the Simplex algorithm is iterated with a Branch-and-Bound algorithm, until the desired optimal solution is found.

OS: DOS, Windows 95

Contact: M.R.C.M. Berkelaar Eindhoven University of Technology Design Automation Section, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands, Phone: +31-40-2474792

Download: ftp://ftp.es.ele.tue.nl/pub/lp—solve

A.2.5.6 LINSOLVE (Non-Commercial Software)

LINSOLVE solves linear programming problems interactively. LINSOLVE can also be used to solve linear GOAL programming problems. The current maximum capacity of a registered program is 120 decision variables, 80 constraints, and 10 objectives. The public domain version will not handle more than 25 decision variables. LINSOLV40.EXE is a 40 column mode of LINSOLVE which can be used with an overhead projector in a classroom setting. You give your problem in an 'as is' facsimile format from a file or keyboard. The program will ask your choices for the options available. You have the choice of maximizing or minimizing, and even printing out the Simplex-tableau should you so wish. LINSOLVE optionally performs a sensitivity analysis on the coefficients of the objective function, and the right-hand sides of the constraints. LINSOLVE applies the two-stage Simplex-method.

OS: DOS

Contact: Timo Salmi, Professor of Accounting and Business Finance University of Vaasa, P.O. Box 297 SF-65101 Vaasa, Finland

Download: ftp://garbo.uwasa.fi/pc/ts/tslin35c.zip

A.2.6 Modeling Languages

A.2.6.1 Modeling Languages Presentation

Modeling systems are designed to help people formulate LPs and analyze their solutions. An LP modeling system takes as input a description of a linear program in a form that people find reasonably natural and convenient, and allows the solution output to be viewed in similar terms; conversion to the forms required by algorithmic codes is done automatically. The collection of statement forms for the input is often called a modeling language.

A.2.6.2 Modeling Languages Products

A.2.6.2.1 AMPL

AMPL, developed at AT&T's Bell Laboratories, is a powerful, yet easy-to-use, modeling environment for problems in linear, non-linear, network and integer programming. Users can formulate optimization models and analyze solutions using common algebraic notation; the computer manages the interface to advanced optimizers. AMPL is a computer language for describing production, distribution, blending, scheduling and many other kinds of problems known generally as large-scale optimization or mathematical programming. AMPL's familiar algebraic notation and interactive command environment are designed to help formulate models, communicate with a wide variety of solvers, and examine solutions.

OS: MS-DOS, Windows 3.1, Windows 95, Windows NT, SunOS 4.1.x (Solaris1.1), Solaris 2.3 or later, UNIX

Vendor: http://www.ampl.com/cm/cs/what/ampl/vendors.html

http://www.ampl.com

A.2.6.2.2 GAMS 2.50

The General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical programming problems. It consists of a language compiler and a stable of integrated high-performance solvers. GAMS is tailored for complex, large scale modeling applications, and allows you to build large maintainable models that can be adapted quickly to new situations.

OS: Windows 95, Windows 98, Windows NT, Windows 2000; UNIX: Sun Solaris, Compaq Digital UNIX, SGI irix, IBM AIX, HP hp/ux, Linux

Vendor: GAMS Development Corporation

http://www.gams.com

A.2.6.2.3 OPL Studio

ILOG OPL Studio delivers what you need to quickly and easily create advanced business optimization solutions. Its interactive graphical environment lets you develop high-level optimization applications without a detailed knowledge of computer programming. ILOG OPL Studio also makes the resources of the ILOG Optimization Suite available to you, letting you harness the power of the world's most advanced optimization tools to improve your enterprise's efficiency -and do it faster. ILOG OPL Studio will save you a lot of development time for your optimization applications, from modeling to deployment.

OS: Windows 95, Windows 98, Windows NT, Sun Solaris

Vendor: ILOG http://www.ilog.com

Demo: http://www.ilog.com/products/oplstudio/trial.cfm

A.2.6.3 Solver Interfaces with AMPL

The following table summarizes availability of solvers for which interfaces to AMPL have been constructed.

Algorithm types listed in the table are distinguished by the problems they solve and the methods they use, as follows:

- **Linear (simplex)**: Linear objective and constraints, by some version of the simplex method.
- **Linear (interior)***:* Linear objective and constraints, by some version of an interior (or barrier) method.
- **Network***:* Linear objective and network flow constraints, by some version of the network simplex method.
- **Quadratic***:* Convex or concave quadratic objective and linear constraints, by either a simplex-type or interior-type method.
- **Non-linear convex**: Convex or concave but not all-linear objective, with linear (and possibly certain nonlinear) constraints, by an interior-type method.
- **Non-linear**: Continuous but not all-linear objective and constraints, by any of several methods including reduced gradient, quasi-newton, augmented lagrangian and interior-point.
- **Complementarity**: Linear or non-linear as above, with additional complementarity conditions.
- **Integer linear**: Linear objective and constraints and some or all integer-valued variables, by a branch-and-bound approach that applies a linear solver to successive sub-problems.
- **Integer non-linear**: Continuous but not all-linear objective and constraints and some or all integer-valued variables, by a branch-and-bound approach that applies a non-linear solver to successive sub-problems.

Table A.1: List of Mathematical Programming Products

Solver	Algorithm Types	Vendor or Download Site
BPMPD	Linear (interior)	Download from Netlib/opt
CONOPT	Non-linear	ARKI Consulting & Development A/S
CPLEX	Linear (simplex) Linear (interior) Network Quadratic Integer linear	ILOG
DONLP2	Non-linear	Download from the DONLP2 site
FortMP	Linear (simplex) Linear (interior) Quadratic Integer linear Integer quadratic	OptiRisk Systems
IPOPT	Non-linear	IPOPT homepage
KNITRO	Non-linear	KNITRO homepage
LAMPS	Linear (simplex) Integer linear	Advanced Mathematical Software
LANCELOT	Non-linear	LANCELOT site
LOQO	Linear (interior) Quadratic Non-linear	R.J. Vanderbei, Princeton Univ: LOQO homepage
LP_SOLVE	Linear (simplex) Integer linear	Download from Michel Berkelaar
MINOS	Linear (simplex) Non-linear	Stanford Business Software, Inc. (see also SOL Optimization Software)
MOSEK	Linear (simplex) Linear (interior) Quadratic, non-linear convex Integer linear Integer quadratic	MOSEK homepage
OSL	Linear (simplex) Linear (interior) Network Quadratic Integer linear	Optimal Solution Technologies (IBM)
PATH	Complementarity	CPNET Computer Sciences Dept, Univ of Wisconsin
SOPT	Linear (simplex) Quadratic Non-linear convex Integer linear	SAITECH
WSAT(OIP)	Integer linear constraints	Download from Joachim P. Walser

XA	Linear (simplex) Integer linear	Sunset Software Technology
XLSOL	Linear (simplex)	Frontline Systems, Inc.
LS-XLSOL	Quadratic Integer linear	
XPRESS	Linear (simplex) Linear (interior) Quadratic Integer linear	Dash Optimization

A.2.7 Stochastic Programming

A.2.7.1 Stochastic Programming Presentation

Stochastic programming supports decision making under uncertainty. It is a methodology for bringing uncertain future scenarios into the traditional decision making framework of linear programming. Just as linear programming models the optimal allocation of constrained resources to meet known demands, stochastic programming models the allocation of today's resources to meet tomorrow's unknown demands in such a way that the user can explore the trade offs with respect to expected risks and rewards and make informed decisions.

A.2.7.1.1 Stochastic Programs

Stochastic programs are mathematical programs where some of the data incorporated into the objective or constraints is uncertain. Uncertainty is usually characterized by a probability distribution on the parameters. Although the uncertainty is rigorously defined, in practice it can range in detail from a few scenarios (possible outcomes of the data) to specific and precise joint probability distributions. The outcomes are generally described in terms of elements w of a set W. W can be, for example, the set of possible demands over the next few months.

When some of the data is random, then solutions and the optimal objective value to the optimization problem are themselves random. A distribution of optimal decisions is generally unimplementable. Ideally, we would like one decision and one optimal objective value.

A.2.7.1.2 Recourse Models

One logical way to pose the problem is to require that we make one decision now and minimize the expected costs (or utilities) of the consequences of that decision. This paradigm is called the recourse model. Suppose x is a vector of decisions that we must take, and y(w) is a vector of decisions that represent new actions or consequences of x. Note that a different set of y's will be chosen for each possible outcome w. The Two-Stage formulation is

Minimize $f_1(x)$ + Expected Value[f_2(y(w), w)]
subject to $g_1(x) <= 0, \ldots g_m(x) <= 0$

h1(x, y(w)) <= 0 for all w in W
...
hk(x, y(w)) <= 0 for all w in W
x in X, y(w) in Y

The set of constraints h_1 ... h_k describe the links between the first stage decisions x and the second stage decisions y(w). Note that we require that each constraint hold with probability 1, or for each possible w in W. The functions f_2 are quite frequently themselves the solutions of mathematical problems. We don't want to make an arbitrary correction (recourse) to the first stage decision; we want to make the best such correction.

Recourse models can be extended in a number of ways. One of the most common is to include more stages. With a multistage problem, we in effect make one decision now, wait for some uncertainty to be resolved (realized), and then make another decision based on what's happened. The objective is to minimize the expected costs of all decisions taken.

A.2.7.2 Applications of Stochastic Programming

Stochastic programming has been applied to a wide variety of areas. Some of the specific problems are part of the Stochastic Programming test set. Other applications are listed below.

- Manufacturing production planning;
- Manufacturing production capacity planning;
- Electrical generation capacity planning;
- Machine scheduling;
- Freight scheduling;
- Dairy farm expansion planning;
- Macroeconomic modeling and planning;
- Timber management;
- Asset liability management;
- Portfolio selection;
- Traffic management;
- Optimal truss design; and
- Automobile dealership inventory management.

A.2.7.3 Stochastic Programming Products IBM Stochastic Solutions

IBM Stochastic Solutions offers new technology to help the user meet the demands of modeling stochastic programs. It has an easy-to-use solver that operates from the command line. It is the first commercial-grade optimization solution to implement the Stochastic Mathematical Programming System (SMPS) input format for multistage stochastic programs. The flexible decomposition solver is robust and fast, and with callable modules, the user has node-by-node access to data and solutions for efficient solution analysis. The components of IBM Stochastic Solutions are a solver, a suite of callable modules for advanced development, and the User's Guide and Reference containing documentation, problem examples, and sample drivers.

OS: Windows 95, Windows NT, AIX, HP-UX, Sun Solaris, SGI IRIX

Vendor: IBM http://www6.software.ibm.com/es/oslv2/startme.htm

A.2.7.3.1 Queue

Queue is an interactive program dedicated to basic queuing models. You can choose between the following queuing models:

M/G/1, D/G/1, M/M/c, M/D/c, D/M/c, G/M/c queues, Finite-Capacity-Queues, Finite-Source-Queues and Transient M/M/1 Queues. (c = number of servers) In several queuing models, Erlangian and Coxian-2 services are assumed for arrivals. The information list defines the Erlangian and the Coaxian-2 distributions in detail: there is therefore also a pedagogical dimension to this program, over and above the simple solution of queuing problems.

OS: DOS

Vendor: Prof. Dr. H.C. Tijms http://www.econ.vu.nl/ectrie/nl/leden/htijms.html

A.2.8 Constraint Satisfaction Problem

A.2.8.1 Constraint Satisfaction Problem Presentation

Constraint Satisfaction Problems (CSP) have been a subject of research in Artificial Intelligence for many years. The pioneering works on networks of constraints were motivated mainly by problems arising in the field of picture processing [Waltz, Montanari]. AI research, concentrated on difficult combinatorial problems, is dated back to sixties and seventies and it has contributed to considerable progress in constraint-based reasoning. Many powerful algorithms were designed that became a basis of current constraint satisfaction algorithms.

A Constraint Satisfaction Problem (CSP) consists of:

- A set of *variables* $X = \{x_1, ..., x_n\}$;
- For each variable x_i, a finite set D_i of possible values (its *domain*); and
- A set of *constraints* restricting the values that the variables can simultaneously take.

Note that values need not be a set of consecutive integers (although often they are), they need not even be numeric.

A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied. We may want to find:

- Just one solution, with no preference as to which one;
- All solutions; and
- An optimal, or at least a good solution, given some objective function defined in terms of some or all of the variables.

Solutions to CSPs can be found by searching systematically through the possible assignments of values to variables. Search methods divide into two broad classes, those that traverse the space of partial solutions (or partial value assignments), and those that explore the space of complete value assignments (to all variables) stochastically.

The reasons for choosing to represent and solve a problem as a CSP rather than, say as a mathematical programming problem are two-fold.

Firstly, the representation as a CSP is often much closer to the original problem: the variables of the CSP directly correspond to problem entities, and the constraints can be expressed without having to be

translated into linear inequalities. This makes the formulation simpler, the solution easier to understand, and the choice of good heuristics to guide the solution strategy more straightforward.

Secondly, although CSP algorithms are essentially very simple, they can sometimes find solution more quickly than if integer programming methods are used.

This tutorial is intended to give a basic grounding in constraint satisfaction problems and some of the algorithms used to solve them. In general, the tasks posed in the constraint satisfaction problem paradigm are computationally intractable (**NP-hard**).

A.2.8.2 Constraint Satisfaction Problem Techniques

A.2.8.2.1 Binarization of Constraints

A constraint can affect any number of variables form 1 to n (n is the number of variables in the problem). It is useful to distinguish two particular cases: unary and binary constraints. Since unary constraints are dealt with by pre-processing the domains of the affected variables, they can be ignored thereafter. If all the constraints of a CSP are binary, the variables and constraints can be represented in a constraint graph and the constraint satisfaction algorithm can exploit the graph search techniques. This is interesting because any constraint can be expressed in terms of binary constraints. Hence, binary CSPs are representative of all CSPs.

A.2.8.2.2 Systematic Search Algorithms

A CSP can be solved using generate-and-test paradigm (GT) that systematically generates each possible value assignment and then it tests to see if it satisfies all the constraints. A more efficient method uses the backtracking paradigm (BT) that is the most common algorithm for performing systematic search. Backtracking incrementally attempts to extend a partial solution toward a complete solution, by repeatedly choosing a value for another variable.

A.2.8.2.3 Consistency Techniques

The late detection of inconsistency is the disadvantage of GT and BT paradigms. Therefore various consistency techniques for constraint graphs were introduced to prune the search space. The consistency-enforcing algorithm makes any partial solution of a small sub-network extensible to some surrounding network. Thus, the inconsistency is detected as soon as possible. The consistency techniques range from simple node-consistency and the very popular arc-consistency to full, but expensive path consistency.

A.2.8.2.4 Constraint Propagation

By integrating systematic search algorithms with consistency techniques, it is possible to get more efficient constraint satisfaction algorithms. Improvements of backtracking algorithm have focused on two phases of the algorithm: moving forward (forward checking and look-ahead schemes) and backtracking (look-back schemes).

A.2.8.2.5 Variable and Value Ordering

The efficiency of search algorithms which incrementally attempts to extend a partial solution depends considerably on the order in which variables are considered for instantiations. Having selected the next variable to assign a value to, a search algorithm has to select a value to assign. Again, this ordering affects the efficiency of the algorithm. There exist various heuristics for dynamic or static ordering of values and variables.

A.2.8.2.6 Reducing Search

The problem of most systematic search algorithms based on backtracking is the occurrence of many "backtracks" to alternative choices which degrade the efficiency of the system. In some special cases, it is possible to completely eliminate the need for backtracking. Also, there exist algorithms which reduce the backtracking by choosing the special variable ordering.

A.2.8.2.7 Heuristics and Stochastic Algorithms

In the last few years, greedy local search strategies became popular, again. These algorithms incrementally alter inconsistent value assignments to all the variables. They use a "repair" or "hill climbing" metaphor to move towards more and more complete solutions. To avoid getting stuck at "local maxima" they are equipped with various heuristics for randomizing the search. Their stochastic nature generally voids the guarantee of "completeness" provided by the systematic search methods.

A.2.8.2.8 Benchmarking and Algorithm Analysis

It is possible to analyze the efficiency of the constraint satisfaction algorithms by traditional approaches that compute the worst-case, average, etc., complexity of the algorithm. However, the methodology for experimental evaluation of the algorithms was also developed. This methodology is based on analyzing populations of randomly-generated binary constraint satisfaction problems and it enables one to analyze the algorithm according to a given class of problems. It also helps to identify where the extremely hard problems occur.

A.2.8.3 Constraint Programming Products

The main contenders are Ilog Solver and CHIP. Ilog has a larger market share, probably because Cosytec were slow to offer CHIP as a \CPP/ library. ICL offer their CHIP-derived system, DecisionPower, to customers only as part of a consultancy package. IF/Prolog is also derived from CHIP, and both offer Boolean, rational, and finite domain (i.e. CSP) solvers. Both CHIP v5 and Ilog benefit from the addition of various sophisticated OR algorithms which allow powerful new symbolic constraints to be offered.

Ilog Solver/Scheduler: The Ilog optimization suite consists of three C++ and JAVA libraries:

- **Ilog Solver** is a basic CSP solver.
- **Ilog Scheduler** is built on top of Solver. It facilitates modelling activities and various resource types, and provides sophisticated algorithms to enforce capacity constraints.
- **Ilog Planner** uses a Simplex-like solver to facilitate production planning.

CHIP v5: Developed by the original Chip team. Prolog version or C library, C++ version available soon. No website, contact: COSYTEC.

IF/Prolog: Developed from Siemens SNI Prolog, hence another from the CHIP family.

Prolog IV: Constraint Solver for rationals, lists, and intervals. Unlike Prolog III which used the non-standard Marseille dialect of Prolog, Prolog IV is ISO compliant.

A.2.8.3.1 CHIP V5

CHIP stands for ``Constraint Handling In Prolog''. It is a new generation logic programming language combining the declarative aspects of Prolog with the efficiency of specialized constraint solving techniques. It brings together techniques from different paradigms in order to allow powerful symbolic

and numerical constraint manipulation. CHIP V5, which is the second generation product in constraint programming technology, allows to develop decision support systems for solving challenging problems, in areas such as production scheduling, logistics or manpower planning. In most practical situations, these problems within a resource optimisation framework, are both very computationally complex and highly evolutive over time.

- **CHIP Technology**

In 1996, COSYTEC launched the second generation of Constraint Programming Tools with CHIP V5. The principal features of this tool – global constraints and open architecture – give COSYTEC a technological advantage acknowledged by the market.

A.2.8.3.2 Global Constraints

COSYTEC was the originator of this concept. It represents a major technological breakthrough, for several reasons:

- This level of abstraction makes it possible to model the problems in a much more concise and more realistic way;
- Calculation times and memory management are dramatically improved;
- Certain types of problem previously considered unsolvable are now solved extremely efficiently (e.g. simultaneous correction of shifts and timetables at LUFTHANSA, nuclear fuel disposal at EDF, procurement logistics for Sun Valley (UK) or scheduling of satellite missions for Alcatel ESPACE);
- Code size is considerably reduced, greatly facilitating evolutionary maintenance of the application; and
- The method for relaxation of a global constraint is defined when the constraint is set, making it possible to solve "over-constrained" problems.

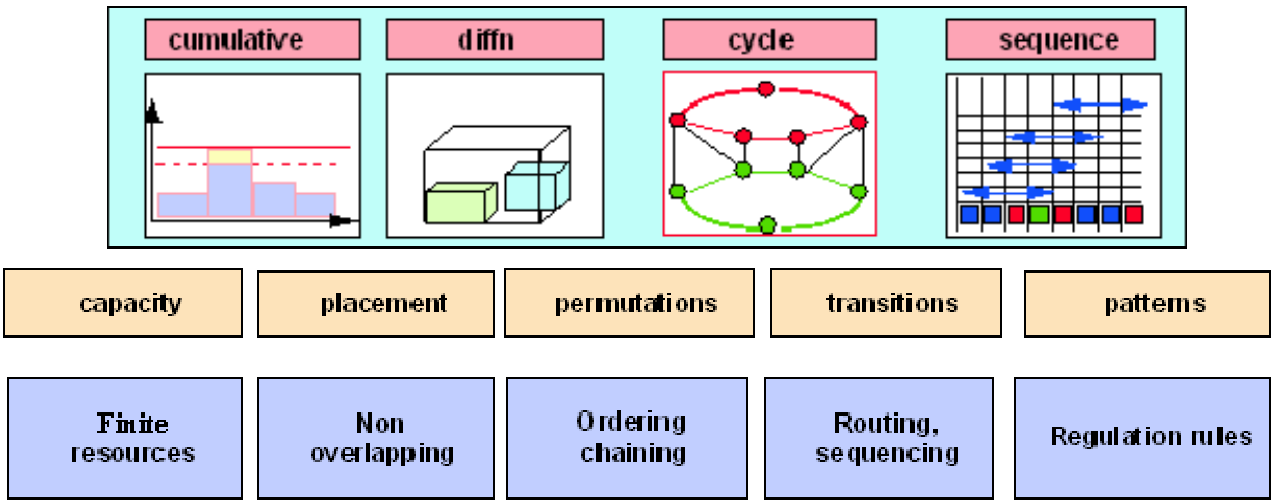

Figure A.2: The New Constraints in CHIP V5 and the Context for their Use.

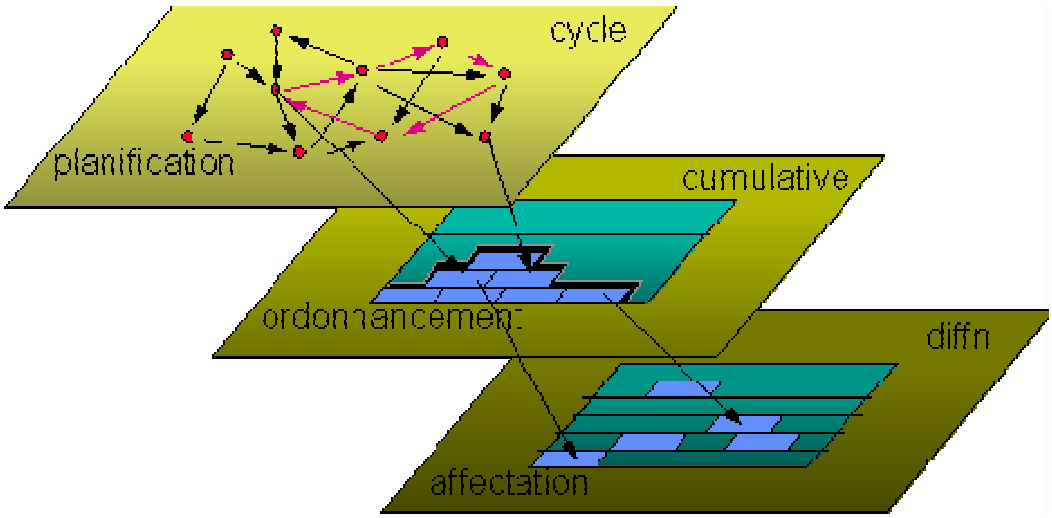

Figure A.3: Crossing of Global Constraints.

The example above illustrates how a concrete problem is represented using global constraints. By crossing them it is possible to maximize deduction and propagation, and quickly converge towards a solution.

For instance, let's imagine a satellite planning problem:

> The **cumulative** constraint is used to model **producer/consumer** behavior, between the energy-producing resources (solar sensors), the storage facility (batteries) and the consumption of this energy; the **diffn** constraint expresses the concept of assignment (a photograph is assigned to a mission, or two photographs of neighboring regions must not overlap); finally, the **cycle** constraint expresses the time and space data concerning the missions (photographs above regions) or the transitions and chaining of activities.

- **Open Architecture**

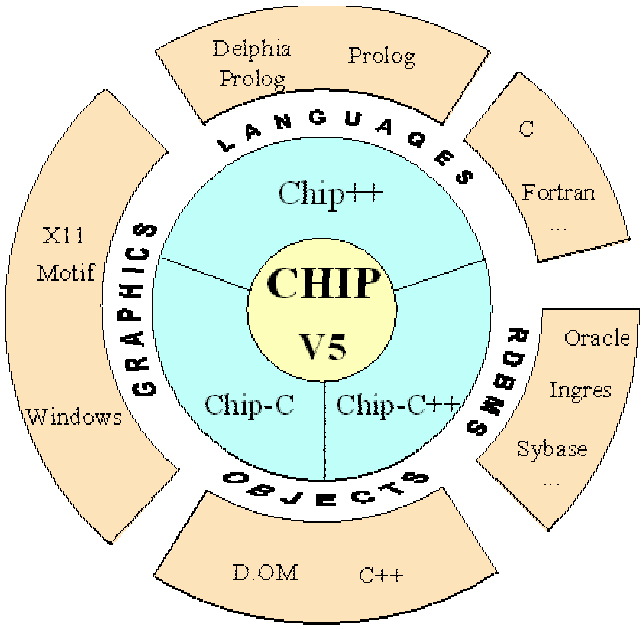

Figure A.4: CHIP Architecture.

CHIP V5's kernel is written entirely in ANSI C. The product's open architecture makes the constraint engine completely independent of the programming language (CHIP-C, CHIP-C++, CHIP++).

RAD tool (Rapid Application Development).

It is often difficult to express real-life optimization, scheduling or planning problems using a standard model, because they are complex and constantly changing.

Constraint Programming tackles and solves this problem efficiently, using an incremental approach in which the constraint model is gradually enriched. Rapid Application Development (RAD) is perfect for designing and validating mathematical models.

However to put this method into action you must have a sophisticated development environment offering fast, user-friendly debugging tools.

The XGIP and GAELL object-oriented graphic tools are linked to the solver. They further facilitate this rapid development process by enabling the developer to display the application's results in graphic form, as interactive Gantt diagrams, tables and curves.

Thanks to this tool, to create a graphic interactive mock-up of the application all you need to do is incorporate the customer's data.

This means a mock-up of the application can be created in a few days, and only two or three weeks are required to create a prototype including the solver.

This type of tool therefore makes it possible to implement a progressive approach during the project comprizing intermediate steps and successive software versions (prototype version 1, 2, etc.). This incremental methodology is ideal for setting up planning and scheduling applications, because this type of project requires user involvement at every stage, from the detailed specifications to development, experimentation and then implementation.

Different solvers included in CHIP.

The CHIP system comprizes two solvers: a solver for rational numbers and a solver on finite domains. These two solvers communicate with each other, enabling Mixed Programming.

The rational number solver in CHIP V5 is a complete solver on continuous domains, based on Gaussian elimination and the simplex method. It uses rational numbers and their exact arithmetic, and functions in incremental mode.

Because its application domain is so large, the solver on finite domains (comprising the global constraints) constitutes the central part of the CHIP V5 system.

A.2.8.3.3 Partial Search Techniques

The Partial Search Techniques recently implemented in CHIP V5 are Meta-Heuristics specially adapted to solving optimization problems. In most cases these techniques aim to find a correct solution every time (if one exists), and in a relatively short time period (e.g. less than one minute). They represent an effective alternative to proximity methods, such as taboo or simulated annealing, without any of the disadvantages (in the case of local optimums).

OS: UNIX, Open-VMS, Windows NT

Vendor: COSYTEC S.A. http://www.cosytec.fr

A.2.8.3.4 ILOG Solver and Scheduler

- **ILOG Solver is ILOG's Constraint-Based Optimization Engine**

A core engine of the ILOG Optimization Suite, this software component provides cutting-edge optimization technology for powering scheduling, sequencing, timetabling, configuration, dispatching and resource-allocation applications with logical constraints.

Available add-ons equip ILOG Solver for tackling a wide range of problems.

What is Constraint Programming? Constraint programming is a programming technology for solving complex combinatorial problems. Data representing a problem are described by domain variables. Each variable has an associated domain, which is the set of its potentially feasible values. Constraints describe the different relationships that must be met within a set of variables in order to solve a problem.

- **ILOG Scheduler Constraint-Based Scheduling**

ILOG Scheduler is a software component that supplements ILOG Solver by providing specialized modeling and algorithmic enhancements for solving problems that involve scheduling tasks over a given period of time:

- Optimizes finite-capacity scheduling problems; and
- Flexible, open and extensible.

A.2.9 Global Optimization

A.2.9.1 Global Optimization Presentation

Global optimization is the task of finding the absolutely best set of parameters to optimize an objective function. In general, there can solutions that are locally optimal but not globally optimal. Consequently, global optimization problems are typically quite difficult to solve exactly; in the context of combinatorial problems, they are often NP-hard. Global optimization problems fall within the broader class of non-linear programming (NLP).

Methods for global optimization problems can be categorized based on the properties of the problem that are used and the types of guarantees that the methods provide for the final solution. A NLP has the form

Minimize	$F(x)$		
subject to	$g_i(x) = 0$	for $i = 1, ..., m_1$	where $m_1 >= 0$
	$h_j(x) >= 0$	for $j = m_{1+1}, ..., m$	where $m >= m_1$

where F is a function of a vector of reals x that is subject to equality and inequality constraints. Some of the most important classes of global optimization problems are differential convex optimization, complementary problems, minimax problems, bi-linear and biconvex programming, continuous global optimization and quadratic programming.

Specific optimization methods have been developed for many classes of global optimization problems. Additionally, general techniques have been developed that appear to have applicability to a wide range of problems.

Combinatorial problems have a linear or non-linear function defined over a set of solutions that is finite but very large. There are a number of significant categories of combinatorial optimization problems, including network problems, scheduling, and transportation. If the function is piecewise linear, the combinatorial problem can be solved exactly with a mixed integer program method, which uses branch and bound. Heuristic methods like simulated annealing, tabu search and genetic algorithms have also been successfully applied to these problems to find approximate solutions.

General unconstrained problems have a non-linear function over reals that is unconstrained (or which have simple bound constraints). A variety of partitioning strategies have been proposed to solve this problem exactly. These methods typically rely on *a priori* knowledge of how rapidly the function can vary (e.g. the Lipshitz constant) or the availability of an analytic formulation of the objective function (e.g. interval methods). Statistical methods also use partitioning to decompose the search space, but they use *a priori* information (or assumptions) about how the objective function can be modeled. A wide variety of other methods have been proposed for solving these problems inexactly, including simulated annealing, genetic algorithms, clustering methods, and continuation methods, which first transform the potential function into a smoother function with fewer local minimizers, and then uses a local minimization procedure to trace the minimizers back to the original function (e.g. Moré and Wu).

General constrained problems have a non-linear function over reals that is constrained. These types of problems have not received as much attention as have general unconstrained problems. However, many of the methods for unconstrained problems have been adapted to handle constraints.

A.2.9.2 Global Optimization Techniques

- Branch and Bound;
- Mixed Integer Programming;
- Interval Methods;
- Clustering Methods;
- Evolutionary Algorithms;
- Hybrid Methods;
- Simulated Annealing;
- Statistical Methods; and
- Tabu Search.

A.2.10 Main Firms in Optimization

A.2.10.1 ILOG

Website: www.ilog.com

ILOG is the leading supplier of advanced software components. ILOG's customizable, pre-built C, C++, and Java components include optimization engines, business rule engines, and interactive user-interface engines. ILOG engines are used to cut development time of mission-critical applications while giving them new capabilities. Top-tier independent software vendors and leading corporations worldwide use ILOG products and services for Web, front-office, and back-office applications. Founded in 1987, ILOG is dually headquartered in Paris, France, and Mountain View, California, with more than 2,000 customers in 30 countries.

A.2.10.1.1 ILOG Product Overview

ILOG makes the world's most advanced software engines, including C++ and Java software components for optimization, visualization and business rules. Leading ISVs and corporations embed ILOG engines within business applications to add key functionality. ILOG technology leadership produces engines that promote faster development of smarter, more intuitive applications. ILOG is the technology and market leader in optimization engines. Major corporations and top ISVs use ILOG optimization engines within

mission-critical applications that ensure the most efficient use of key resources like people, equipment, inventory and time.

- **ILOG Visualization Suite**

ILOG is the technology and market leader in advanced user-interface engines. ILOG user-interface engines are embedded in applications that manage complex systems – such as telecom networks or supply chains – as well as in distributed applications like workflow. ILOG technology speeds development time by up to 80%, and greatly increases user productivity.

- **ILOG Rules**

ILOG is the technology leader in high-performance rule engines that automate decision-making within many e-business applications. In today's constantly changing, short-turnaround Web environments, using ILOG components ensures that applications can adapt quickly, evolving ahead of the competition. ILOG Optimization SuiteILOG supplies the world's most powerful and comprehensive components for developing optimization applications.

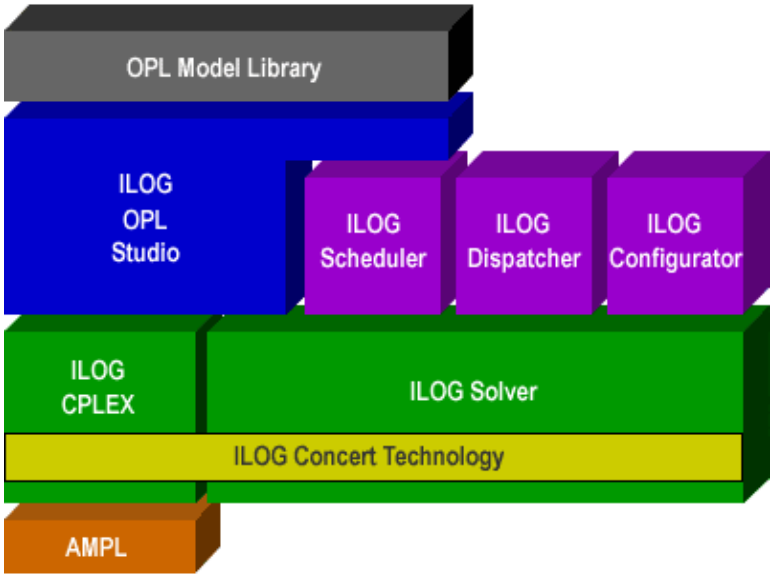

Figure A.5: ILOG Optimization Suite.

- **ILOG Optimization Suite**

Core optimization engines:

- ILOG CPLEX Mathematical programming engine
- ILOG Solver Constraint programming engine
- ILOG Concert Technology: Both ILOG CPLEX and ILOG Solver include ILOG Concert Technology

Vertical engine extensions:

- ILOG SchedulerConstraint-based scheduling
- ILOG DispatcherVehicle routing and personnel dispatching
- ILOG ConfiguratorConfiguration of products and services

Modeling tools:

- ILOG OPL StudioModeling environment for rapid development of optimization applications
- ILOG OPL Model LibraryOPL models to jump-start development
- AMPL CPLEX SystemModeling support integrated with ILOG CPLEX

- **ILOG CPLEX**

World's Leading Mathematical Programming Optimizers: ILOG CPLEX delivers high-performance, robust, flexible optimizers for solving linear, mixed-integer and quadratic programming problems in mission-critical resource allocation applications. Virtually every leading end-user and software provider in supply chain planning, telecommunication network design and transportation logistics depends on the unequaled solving power of ILOG CPLEX.

Robust algorithms for demanding problemsILOG pioneered development of the world's fastest, most robust implementations of the fundamental algorithms needed to solve today's most demanding mathematical optimization problems. ILOG CPLEX has solved problems with millions of constraints and variables, and consistently sets new records for mathematical programming software performance. Leading hardware vendors use ILOG CPLEX to measure the computational performance of their latest CPUs.

Industry-leading supportNo other vendor can match ILOG's reputation in the industry for performance, reliability, flexibility and support. ILOG's ongoing commitment to pushing the performance envelope ensures ILOG customers that their investment in ILOG CPLEX is protected well into the future.

Features

High performanceILOG CPLEX optimizers deliver the speed needed to solve huge, real-world optimization problems, as well as the quick solutions that e-commerce applications require. ILOG CPLEX is the de facto standard for mathematical optimization, with thousands of users worldwide.

Robust and reliableILOG CPLEX has been fully proven in the most demanding real-world applications. Only ILOG CPLEX provides the accuracy and dependability that key business applications demand.

Flexible ILOG CPLEX Component Libraries empower developers with the ability to embed the ILOG CPLEX engine seamlessly and efficiently in their applications. Only ILOG CPLEX provides the full range of functionality and flexibility that optimization applications require. ILOG CPLEX can be accessed from most programming environments and used on a wide variety of platforms, providing true portability.

Leaders in supply chain planning, telecommunication network design and transportation logistics depends on ILOG CPLEX optimizers for solving linear, mixed-integer and quadratic programming problems in resource-allocation applications.

- **ILOG Solver Constraint Programming Engine**

ILOG Solver is ILOG's constraint-based optimization engine.

A core engine of the ILOG Optimization Suite, this software component provides cutting-edge optimization technology for powering scheduling, sequencing, timetabling, configuration, dispatching and resource-allocation applications with logical constraints.

Available add-ons equip ILOG Solver for tackling a wide range of problems.

What is Constraint Programming? Constraint programming is a programming technology for solving complex combinatorial problems. Data representing a problem are described by domain variables. Each variable has an associated domain, which is the set of its potentially feasible values. Constraints describe the different relationships that must be met within a set of variables in order to solve a problem.

- **ILOG Scheduler Constraint-Based Scheduling**

ILOG Scheduler is a software component that supplements ILOG Solver by providing specialized modeling and algorithmic enhancements for solving problems that involve scheduling tasks over a given period of time:

- Optimizes finite-capacity scheduling problems; and
- Flexible, open and extensible.

Quick, intuitive modelingILOG Scheduler's modeling objects lets users quickly and intuitively model resources and activities, as well as temporal and capacity constraints. Its state-of-the-art algorithms rapidly and accurately compute schedules that satisfy business objectives and constraints.

Increased efficiency, reduced cost Scheduling is at the center of many strategic corporate problems, and is often the area where companies find the greatest savings. ILOG Scheduler users typically report cost savings of 20%, and a 75% reduction in operational planning time. Application development is also shortened dramatically, thanks to ILOG Scheduler's powerful, easy-to-use application program interface (API).

- **ILOG OPL Studio Rapid Development and Deployment for Optimization Applications**

ILOG OPL Studio is a complete platform for leveraging valuable resources with optimization technology. Powered by ILOG's leading optimization engines, this comprehensive modeling system expedites development and deployment.

Fast development Minimal computer programming is required to develop optimization applications, thanks to ILOG OPL Studio's intuitive optimization programming language (OPL):

- Reduce development time for a wide range of optimization problems, ranging from short-term operational scheduling to long-term economic planning;
- Experiment with both constraint programming and mathematical programming to determine the most effective method for solving optimization problems; and
- Descriptive OPL syntax produces substantially simpler code than traditional programming languages, reducing development time from weeks to days.

Smooth deployment Optimization projects advance directly from concept to implementation, drawing on OPL Component Libraries:

- Embed OPL models directly into business applications after analysis and testing;
- Connect optimization systems directly to data sources, reading business information and storing optimized solutions; and
- Solve large or complex optimization problems, utilizing support for powerful computation servers.

A.2.10.2 Dash Optimization

Website: www.dashoptimization.com

Dash develops and markets Xpress-MP, the world's leading software product for modeling and optimization. Xpress-MP solves large-scale optimization problems and enables better business decisions and resulting financial benefits in areas such as supply chain management, operations, logistics and asset management. It has been applied in sectors as diverse as manufacturing, processing, distribution, retailing, transport, finance and investment.

Dash is completely focused on optimization software and works in close partnership with its customers and partners, enabling them to get the best possible performance from Xpress-MP. Xpress-MP is embedded in many software products and solutions as a component, making leading edge optimization accessible to a wide range of customers and applications. Through expertise in the product and its application, excellent customer support and fast track product development, Dash continues to maintain Xpress-MP at the cutting edge.

A.2.10.2.1 Dash Optimization Products

- Xpress-MP – the fast track from formulation to solution
- Xpress-MP is the preferred choice for end-user applications
- Powerful yet flexible model building tools
- The fastest optimizers around
- Able to solve large problems
- Rapid development
- Xpress-MP – the superior optimization software component
- Xpress-MP is the preferred optimization component for software product developers
- Easy to integrate and Reliable
- Powerful, flexible, industry standard interfaces
- Delivers state-of-the-art technology

The Xpress-Optimizer

The Xpress-Optimizer features three optimization algorithms which enable the user to solve linear programming problems (LP), mixed integer programming problems (MIP), quadratic programming problems (QP), and mixed integer quadratic programming problems (MIQP).

The simplex optimizer, which includes primal and dual methods, solves LP problems, and is also used within a branch-and-bound framework to solve MIP and MIQP problems**. The Newton barrier optimizer** is an interior point method for solving LP and QP problems. Xpress-MP uses ultra-efficient sparse matrix handling to allow it to solve the largest problems in record time. A presolve procedure reduces the size of the problem before it is solved, sometimes by an order of magnitude. Xpress-MP is also noted for its ability to solve numerically hard or unstable problems, which is one of the reasons Xpress-MP is the clear market leader in the process industries.

The MIP/MIQP optimizer uses a sophisticated branch-and-bound algorithm to solve MIP and MIQP problems, and is particularly known for its ability to find high quality solutions fast. MIP problems can have an exponential number of possible solutions, and the essential property of the Xpress MIP optimizer

is its ability to cut down the number of solutions to a manageable size, and then to navigate through them so it can find good ones quickly.

Some of the more sophisticated techniques include various classes of cutting planes, which are generated automatically during the optimization to improve the quality of bounds and reduce the size of the search (so the MIP algorithm is really called "branch-cut-cut"). The presolve is particularly effective on MIP problems, as it is able to tighten the formulation, which improves the quality of initial solutions and enables better cutting planes to be generated.

The Hyper32 edition of Xpress-MP has no internal limits on problem size, allowing the user to solve any problem which can physically be accommodated within the limits of 32-bit memory on their computer. And if that isn't big enough, Hyper64 unleashes the power of 64-bit computing. Two restricted size editions, Extended and Professional, are also available. When it is important to solve MIP problems in the shortest possible time, or to obtain solutions for the hardest MIP problems, Xpress-Parallel is the ideal solution. Operating on multi-processor machines or across a network of computers, it enables the user to harness all of the computing power at their disposal to solve MIP problems in parallel.

A.2.10.2.2 Xpress-MP Architecture

Xpress-MP is a suite of optimization software, used to solve optimization problems. It is used in three different ways:

- As a component by ISVs to embed optimization functionality within their own products;
- By consultants to offer optimization solutions to their clients; and
- By business analysts and other end-users within large organizations to solve their own optimization problems directly.

It is used by thousands of companies worldwide to solve all sorts of problems, ranging from supply chain planning to portfolio management, from process industry refining to e-commerce.

Xpress-MP features:

- Different optimization algorithms for solving different problem types;
- Different capacities for solving problems of various sizes;
- Different modeling interfaces which allow the user to define their problem in different ways various software products and components, suitable for using Xpress-MP in an embedded system or as a stand-alone application; and
- Available on all common computer platforms.

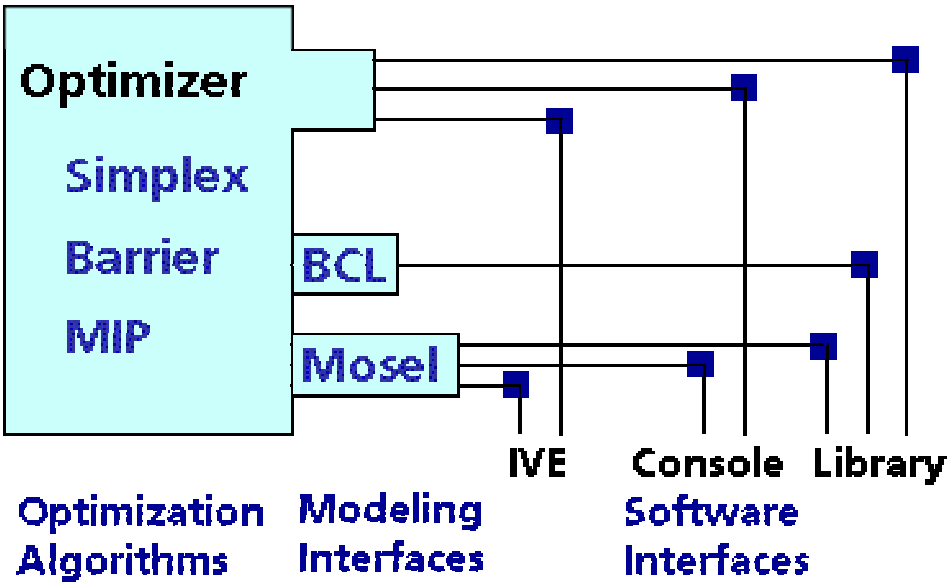

Figure A.6: Xpress-MP Architecture.

A.2.11 Decision Theoretic Approaches

A.2.11.1 Multi Criteria Analysis

- **MCA Introduction**

There are a number of methods which have been developed to provide a rational model for making decisions. The purpose of these techniques is to provide a decision maker with a rational perspective for selecting the "best" path. These techniques help a decision maker frame the known criteria and alternatives and deal with uncertainty where it may be present. The term for the body of these techniques is Decision Analysis (DA). These techniques are higher level of analysis than those oriented exclusively toward finance such as return on investment (ROI) or payback period. The objective of development in decision analysis has been to improve the ability of the human decision maker to make more timely and better quality decisions. Toward this end, extensive algorithmic techniques have been developed for properly framing the decision environment. Research has shown that the more sophisticated techniques tend to produce results that decision makers prefer – but those same techniques require considerable time to understand and to utilize. Therefore they are under utilized. At the same time education in this area needs to be expanded. Many decision makers continue to rely upon faulty, personal, inductive methods and are unaware of the efficacy of decision science methods. Fortunately packaged software is now available to help in applying these methods and for easily gathering opinion and for weighing alternatives.

- **Multi Criteria Decision Analysis Methodology**

A decision analysis technique may contain some or all of the following elements.

Table A.2: Decision Analysis Checklist

Element	Purpose is to:
Goal/Challenge	Succinctly state the ultimate end point of the challenge
Alternatives/Options/Objectives	Help in eliciting and describing the list of alternative measures or objectives to be met in attacking the challenge
Criteria/Objectives/Attributes	List such items as cost, technical, social, and political factors to be considered in the decision
Uncertainty/Preference	Provide a method for attaching information relating to the probability or expected value of an occurrence
Measurement Scales	Specifically value a criteria or alternative
Synthesis Technique	Provide for combining all the elicited and evaluated data to produce a specific answer or ranking in the alternatives
Decision Makers	Poll and apply values from multiple participants

- **MCA Techniques**

A.2.11.1.1 The Performance Matrix

A standard feature of multi-criteria analysis is a performance matrix, or consequence table, in which each row describes an option and each column describes the performance of the options against each criterion. The individual performance assessments are often numerical, but may also be expressed as 'bullet point' scores, or colour coding.

A.2.11.1.2 Scoring and Weighting

MCA techniques commonly apply numerical analysis to a performance matrix in two stages:

- **Scoring:** the expected consequences of each option are assigned a numerical score on a strength of preference scale for each option for each criterion. More preferred options score higher on the scale, and less preferred options score lower. In practice, scales extending from 0 to 100 are often used, where 0 represents a real or hypothetical least preferred option, and 100 is associated with a real or hypothetical most preferred option. All options considered in the MCA would then fall between 0 and 100.
- **Weighting:** numerical weights are assigned to define, for each criterion, the relative valuations of a shift between the top and bottom of the chosen scale.

A.2.11.1.3 Direct Analysis of the Performance Matrix

A limited amount of information about options' relative merits can be obtained by direct inspection of the performance matrix. An initial step can be to see if any of the options are dominated by others.

Dominance occurs when one option performs at least as well as another on all criteria and strictly better than the other on at least one criterion. In principle, one option might dominate all others, but in practice this is unlikely. When it does occur, it is helpful to ask if there is some advantage of the dominated option that is not represented by the criteria; this may reveal new criteria that have been overlooked. Dominance is more likely just to enable the decision-making team to eliminate dominated options from further consideration.

Once any dominance analysis has been concluded, the next stage is for the decision-making team to determine whether trade-offs between different criteria are acceptable, so that good performance on one criterion can in principle compensate for weaker performance on another. Most public decisions admit such trade-offs, but there may be some circumstances, perhaps where ethical issues are central, where trade-offs of this type are not acceptable. If it is not acceptable to consider trade-offs between criteria, then there are a limited number of non-compensatory MCA techniques available.

Where compensation is acceptable, most MCA methods involve implicit or explicit aggregation of each option's performance across all the criteria to form an overall assessment of each option, on the basis of which the set of options can be compared. The principal difference between the main families of MCA methods is the way in which this aggregation is done.

A.2.11.1.3.1 Multi-Attribute Utility Theory

There is no normative model of how individuals should make multi-criteria choices that is without critics. The one that comes closest to universal acceptance is based on multi-attribute utility theory and derives from the work of von Neumann and Morgenstern, and of Savage, in the 1940s and 1950s.

While this work provided powerful theoretical insights, it does not directly help decision makers in undertaking complex multi-criteria decision tasks. The breakthrough in this respect is the work of Keeney and Raiffa, published in 1976. They developed a set of procedures, consistent with the earlier normative foundations, which would allow decision makers to evaluate multi-criteria options in practice.

There are three building blocks for their procedures. First is the performance matrix and the second is procedures to determine whether criteria are independent of each other or not. The third consists of ways of estimating the parameters in a mathematical function which allow the estimation of a single number index, U, to express the decision maker's overall valuation of an option in terms of the value of its performance on each of the separate criteria.

The Keeney and Raiffa approach to decision support has been applied to many real decisions, in both the private and public sectors. Although well-regarded and effective, in its most general form it is relatively complex and best implemented by specialists on major projects where time and expertise are both necessary and available.

What makes the Keeney and Raiffa model potentially demanding to apply is firstly that it takes uncertainty formally into account, building it directly into the decision support models and secondly that it allows attributes to interact with each other in other than a simple, additive fashion. It does not assume mutual independence of preferences. In certain circumstances, it can be important to build into the analysis one or both of these factors, but often in practice it may be better to ignore them in order to allow a simpler and more transparent decision support to be implemented more quickly, by a wider range of users and for a larger set of problem types.

A.2.11.1.3.2 Linear Additive Models

If it can either be proved, or reasonably assumed, that the criteria are preferentially independent of each other and if uncertainty is not formally built into the MCA model, then the simple linear additive evaluation model is applicable. The linear model shows how an option's values on the many criteria can be combined into one overall value. This is done by multiplying the value score on each criterion by the weight of that criterion, and then adding all those weighted scores together. However, this simple arithmetic is only appropriate if the criteria are mutually preference independent. Most MCA approaches use this additive model, and it is the basis of the MCDA model.

Models of this type have a well-established record of providing robust and effective support to decision-makers working on a range of problems and in various circumstances. They will form the foundation for the more detailed recommendations we shall give later.

However, as was argued earlier, the variety of circumstances in which decision support has been sought has led to the development of a range of different decision support models. A number of these will now be described and related to the basic MCDA model.

A.2.11.1.3.3 The Analytical Hierarchy Process

We will describe in some detail appropriate methods to assess the scores that are the basis of the performance matrix and for judging the weights in the linear additive model.

The Analytic Hierarchy Process (AHP) also develops a linear additive model, but, in its standard format, uses procedures for deriving the weights and the scores achieved by alternatives which are based, respectively, on pairwise comparisons between criteria and between options. Thus, for example, in assessing weights, the decision maker is asked a series of questions, each of which asks how important one particular criterion is relative to another for the decision being addressed.

The strengths and weaknesses of the AHP have been the subject of substantial debate among specialists in MCA. It is clear that users generally find the pairwise comparison form of data input straightforward and convenient. This feature is exploited in MCDA by the MACBETH approach to scoring and weighting and the REMBRANDT approach. On the other hand, serious doubts have been raised about the theoretical foundations of the AHP and about some of its properties. In particular, the rank reversal phenomenon has caused concern This is the possibility that, simply by adding another option to the list of options being evaluated, the ranking of two other options, not related in any way to the new one, can be reversed. This is seen by many as inconsistent with rational evaluation of options and thus questions the underlying theoretical basis of the AHP.

A.2.11.1.3.4 Outranking Methods

A rather different approach from any of those discussed so far has been developed in France and has achieved a fair degree of application in some continental European countries. It depends upon the concept of outranking. The methods that have evolved all use outranking to seek to eliminate alternatives that are, in a particular sense, ‘dominated’. However, unlike the straightforward dominance idea, dominance within the outranking frame of reference uses weights to give more influence to some criteria than others.

One option is said to outrank another if it outperforms the other on enough criteria of sufficient importance (as reflected by the sum of the criteria weights) and is not outperformed by the other option in the sense of recording a significantly inferior performance on any one criterion. All options are then assessed in terms of the extent to which they exhibit sufficient outranking with respect to the full set of options being considered as measured against a pair of threshold parameters. An explanation of precisely how outranking can identify a preferred option, or a set of preferred options for further investigation.

An interesting feature of outranking methods is that it possible, under certain conditions, for two options to be classified as ‘incomparable’ (‘difficult to compare’ is probably a better way to express the idea). Incomparability of two options is not the same as indifference between two options and might, for example, be associated with missing information at the time the assessment is made. This is not an unlikely occurrence in many decision making exercises. Building this possibility into the mathematical structure of outranking allows formal analysis of the problem to continue while neither imposing a judgement of indifference which cannot be supported nor dropping the option entirely, simply because information is not to hand.

The main concern voiced about the outranking approach is that it is dependent on some rather arbitrary definitions of what precisely constitutes outranking and how the threshold parameters are set and later manipulated by the decision maker.

The outranking concept does, however, indirectly capture some of the political realities of decision making. In particular it downgrades options that perform badly on any one criterion (which might in turn activate strong lobbying from concerned parties and difficulty in implementing the option in question). It can also be an effective tool for exploring how preferences between options come to be formed.

A.2.11.1.3.5 Procedures that Use Qualitative Data Inputs

The view taken in this manual is that reliable and transparent support for decision making is usually best achieved using numerical weights and scores on a cardinal scale. There are some exceptions, for example application of dominance and use of models that approximate the linear additive model but are based on ranking of weights. However, in general, it is a fair generalisation that the less precise the data inputs to any decision support procedure, the less precise and reliable will be the outputs that it generates.

Nonetheless, it is the case that decision makers working in government are frequently faced with circumstances where the information in the performance matrix, or about preference weights, consists of qualitative judgements. A number of methods exist to respond to this.

One group revolves around approximation to the linear additive model. In this respect they are relatively transparent, although they may involve significant amounts of data processing and, consistent with the fact that imprecise inputs rarely generate precise outputs to appraisal processes, usually require some extra assumptions to be made if, say, a single preferred option is to be identified, or even a ranking of options.

An alternative approach, largely developed in the Netherlands, has instead sought to develop procedures which amend outranking models in order to allow them to process imprecise, qualitative data inputs. They share many of the characteristics of (cardinal scale) outranking methods and have achieved only a limited degree of application, most often in urban and regional planning.

A.2.11.1.4 MCA Methods Based on Fuzzy Sets

A different response to the imprecision that surrounds much of the data on which decision making is based has been to look to the newly developing field of fuzzy sets to provide a basis for decision making models. However, methods of this type are not yet widely applied.

Fuzzy sets attempt to capture the idea that our natural language in discussing issues is not precise. Options are ‘fairly attractive’ from a particular point of view or ‘rather expensive’, not simply ‘attractive’ or ‘expensive’. Fuzzy arithmetic then tries to capture these qualified assessments using the idea of a membership function, through which an option would belong to the set of, say, ‘attractive’ options with a given degree of membership, lying between 0 and 1.

Building on assessments expressed in this way, fuzzy MCA models develop procedures for aggregating fuzzy performance levels using weights that are sometimes also represented as fuzzy quantities. However, these methods tend to be difficult for non-specialists to understand, do not have clear theoretical foundations from the perspective of modelling decision makers’ preferences and have not yet established that they have any critical advantages that are not available in other, more conventional models.

A.2.11.1.4.1 Other MCA Methods

The preceding sections have outlined some of the main types of MCA model that have been proposed as potentially applicable to public sector decision making. There are many others, some of which have a

record of application, but many others which have not advanced significantly beyond the conceptual phase. Categories that have not been explicitly discussed but which are referred to in the MCA literature include methods based on Rough Sets, or on Ideal Points and several methods that are heavily dependent on interactive development, using specially constructed computer packages.

A.2.11.2 Decision Support System

A.2.11.2.1 Decision Support Systems Presentation

Decision Support Systems are computer-based systems that help decision makers make better decisions. DSS support individual and group, one-time or re-occurring, decisions that address semi-structured and unstructured problems requiring human judgement. DSS are interactive systems that incorporate data and models. Decision makers and professionals at all levels of the organization use one or more types of DSS including Ad-hoc DSS, Organizational DSS, Group DSS, and Executive Information System.

A.2.11.2.2 Decision Support Systems and Multi Criteria Decision Analysis Products

A.2.11.2.2.1 ELECTRE IS

ELECTRE IS is a generalization of ELECTRE I. It is a multicriteria method which enables to use pseudo-criteria (criteria with thresholds). Given a finite set of alternatives evaluated on a consistent family of criteria, ELECTRE IS supports the user in the process of selecting one alternative or a subset of alternatives. The method consists of two parts:

- Construction of one crisp outranking for modelling the decision-maker's preferences; and
- Exploitation of the graph corresponding to this relation.

The subset searched is the kernel of the graph.

Information: www.dauphine.lamsade.fr

A.2.11.2.2.2 ELECTRE II-IV

ELECTRE III starts with a finite set of actions evaluated on a consistent family of pseudo-criteria and aggregates these partial preferences into a fuzzy outranking relation. ELECTRE IV builds several non-fuzzy outranking relations when criteria cannot be weighted.

Two complete pre-orders are then obtained through a "distillation" procedure, either from the fuzzy outranking relation of ELECTRE III, or from the non-fuzzy outranking relations provided by ELECTRE IV. The intersection of these pre-orders indicates the most reliable part of the global preference.

Information: www.dauphine.lamsade.fr

A.2.11.2.2.3 ELECTRE TRI

ELECTRE TRI is a multiple criteria decision aiding tool designed to deal with sorting problems (segmentation problems). A segmentation problem consist in analysing the intrinsic value of each alternative (file, candidate, project, ...) in order to propose an appropriate recommendation for each of them.

Starting from a finite set of alternative evaluated on a family of quantitative and/or qualitative criteria and from a set of ordered categories corresponding to pre-defined recommendations (for example: very good,

good,… , mediums, bad), ELECTRE TRI provides the users two different procedures that assign each alternative to the categories.

In opposition with the classical procedures based on the weighted sum (possibility of compensation), the two proposed procedures in the ELECTRE TRI method refuse such possibility of total compensation among the evaluations of alternatives according to the various criteria. The assignment of an alternative is grounded on the comparison of the alternative to reference alternatives by mean of an outranking relation. These two procedures differ according to their behaviour (either pessimistic or optimistic) on the way to deal with situations in which certain alternatives are incomparable with some reference alternatives.

ELECTRE TRI Assistant is a new module in the software that enables the user to calibrate the ELECTRE TRI model indirectly, i.e. fix the model parameters by giving some examples of assignments (corresponding to desired assignments or past decisions). The values of parameter are inferred through a certain form of regression on assignment examples. Hence, ELECTRE TRI Assistant reduces the cognitive effort required from the decision maker to elicit the preferential parameters.

Information: www.dauphine.lamsade.fr

A.2.11.2.2.4 UTA PLUS

The method can be used to solve the problems of multicriteria choice and ranking on a set A of alternatives. It constructs an additive utility function from a preference weak order defined by the user on a subset Aþ of reference alternatives. The construction, based on a principle of ordinal regression, consists of solving a small LP-problem. The software proposes marginal utility functions in piecewise linear form as compatible as possible with the given weak order. It allows the user to modify interactively the marginal utility functions within limits following from a sensitivity analysis of the ordinal regression problem. For these modifications, the user is helped by a friendly graphical interface.

Information: www.dauphine.lamsade.fr

A.2.11.2.2.5 Analytica

Analytica is a visual software tool for creating, analyzing, and communicating quantitative models. Analytica provides an alternative to the spreadsheet, providing graphical influence diagrams to show qualitative structure of models, hierarchical models to organize complicated models into manageable modules, and intelligent arrays with the power to scale simple models up to handle large problems.

OS: Windows 95, Windows 98, Windows NT 4.0, Macintosh OS

Vendor: Lumina Decision Systems Inc. http://www.lumina.com

Demo: http://www.lumina.com/reg/trialanalytica.htm

A.2.11.2.2.6 Criterium DecisionPlus

Criterium DecisionPlus supports users in structuring and analyzing complex decisions between alternatives and involving multiple criteria.

Two leading methodologies for multi-criteria analysis are combined with uncertainty analysis: the Analytic Hierarchy Process and Simple Multi-Attribute Rating Technique. Based on these methodologies, the software aids the decision maker in consistently assigning relative importance to criteria, and rating alternatives against those criteria. The weighted score of each alternative against those criteria indicates how

well it meets the decision criteria. Graphic analysis windows aid the user in analyzing the decision for reasonableness, robustness and tradeoffs.

Minimum requirements may be captured using the software's rule syntax, e.g. "applicants experience must be at least 3 years". By assigning probability distributions to capture uncertain data, the user knows not only which alternative best meets their criteria, but how likely that alternative is to be truly the best choice. The outcomes based on the assigned uncertainties are overlaid on the results based on the single-valued estimates to give a powerful visual representation of the impact of uncertain data on the decision process. The software analyzes which information's uncertainty should be reduced to best clarify the decision.

OS: Windows 3.1, Windows 95, Windows 98, Windows NT

Vendor: InfoHarvest, Inc. http://www.infoharvest.com

Demo: http://www.infoharvest.com/infoharv/CDPFreeDownloads.htm

A.2.11.2.2.7 DPL

Key features: DPL (Decision Programming Language) is decision analysis software developed to meet the requirements of decision-makers in business and government. DPL offers an advanced synthesis of the two major decision-making tools, influence diagrams and decision trees, which assist in structuring complete and focused analyses. DPLs powerful solution algorithms and many graphical outputs provide comprehensive and insightful results. DPL is currently being used by over 400 companies, government agencies, universities and research institutes in 31 countries.

OS: Windows 95, Windows NT

Vendor: Applied Decision Analysis. http://www.adainc.com

A.2.11.2.3 Expert Choice

Expert Choice for Windows is a multi-criteria decision support software tool based on the world's most popular decision-making methodology: the Analytic Hierarchy Process (AHP). With Expert Choice, defining your goals, identifying the criteria and alternatives, and evaluating key tradeoffs is a straight-forward and thorough process. It assists you in building a model for your decision, then leads you in judging, via pair-wise comparisons, the relative importance of the variables. Expert Choice then synthesizes your judgments to arrive at a conclusion and allows you to examine how changing the weighting of your criteria affects your outcome.

OS: Windows 3.11, Windows 95, Windows 98, Windows NT

Vendor: Palisade Corporation http://www.palisade.com

A.3 SIMULATION

Simulation is a powerful tool for those who want to analyze, design, and operate complex systems. It lets users create models of real-world processes which are too complex to be analyzed by spreadsheets or flowcharts. It is a cost-effective means of exploring new processes, without having to resort to pilot programs. And it is an efficient communication tool, showing how an operation works while stimulating creative thinking about how it can be improved. Simulation is used in industry, government, and educational institutions to shorten the design cycle, reduce costs, and enhance knowledge.

A.3.1 Introduction

Computer system users, administrators, and designers usually have a goal of highest performance at lowest cost. Modeling and simulation of system design trade off is good preparation for design and engineering decisions in real world jobs.

A computer simulation can be effectively used in a particular scenario. In addition to its use as a tool to better understand and optimize performance and/or reliability of systems, simulation is also extensively used to verify the correctness of designs. Most if not all digital integrated circuits manufactured today are first extensively simulated before they are manufactured to identify and correct design errors. Simulation early in the design cycle is important because the cost to repair mistakes increases dramatically the later in the product life cycle that the error is detected. Another important application of simulation is in developing "virtual environments", e.g. for training. Analogous to the holodeck in the popular science-fiction television program Star Trek, simulations generate dynamic environments with which users can interact "as if they were really there." Such simulations are used extensively today to train military personnel for battlefield situations, at a fraction of the cost of running exercises involving real tanks, aircraft, etc.

Dynamic modeling in organizations is the collective ability to understand the implications of change over time. This skill lies at the heart of successful strategic decision process. The availability of effective visual modeling and simulation enables the analyst and the decision-maker to boost their dynamic decision by rehearsing strategy to avoid hidden pitfalls.

System Simulation is the mimicking of the operation of a real system, such as the day-to-day operation of a bank, or the value of a stock portfolio over a time period, or the running of an assembly line in a factory, or the staff assignment of a hospital or a security company, in a computer. Instead of building extensive mathematical models by experts, the readily available simulation software has made it possible to model and analyze the operation of a real system by non-experts, who are managers but not programmers.

A simulation is the execution of a model, represented by a computer program that gives information about the system being investigated. The simulation approach of analyzing a model is opposed to the analytical approach, where the method of analyzing the system is purely theoretical. As this approach is more reliable, the simulation approach gives more flexibility and convenience. The activities of the model consist of events, which are activated at certain points in time and in this way affect the overall state of the system. The points in time that an event is activated are randomized, so no input from outside the system is required. Events exist autonomously and they are discrete so between the execution of two events nothing happens.

In the field of simulation, the concept of "principle of computational equivalence" has beneficial implications for the decision-maker. Simulated experimentation accelerates and replaces effectively the "wait and see" anxieties in discovering new insight and explanations of future behavior of the real system.

A.3.2 Topics in Descriptive Simulation Modeling

A.3.2.1 Modeling and Simulation

Simulation in general is to pretend that one deals with a real thing while really working with an imitation. In operations research the imitation is a computer model of the simulated reality. A flight simulator on a PC is also a computer model of some aspects of the flight: it shows on the screen the controls and what the "pilot" (the youngster who operates it) is supposed to see from the "cockpit" (his armchair).

A.3.2.1.1 Why to Use Models?

To fly a simulator is safer and cheaper than the real airplane. For precisely this reason, models are used in industry commerce and military: it is very costly, dangerous and often impossible to make experiments with real systems. Provided that models are adequate descriptions of reality (they are valid), experimenting with them can save money, suffering and even time.

A.3.2.1.2 When to Use Simulation?

Systems that change with time, such as a gas station where cars come and go (called dynamic systems) and involve randomness. Nobody can guess at exactly which time the next car should arrive at the station, are good candidates for simulation. Modeling complex dynamic systems theoretically need too many simplifications and the emerging models may not be therefore valid. Simulation does not require that many simplifying assumptions, making it the only tool even in absence of randomness.

A.3.2.1.3 How to Simulate?

Suppose we are interested in a gas station. We may describe the behavior of this system graphically by plotting the number of cars in the station; the state of the system. Every time a car arrives the graph increases by one unit while a departing car causes the graph to drop one unit. This graph (called sample path), could be obtained from observation of a real station, but could also be artificially constructed. Such artificial construction and the analysis of the resulting sample path (or more sample paths in more complex cases) consists of the simulation.

Types of Simulations:

- **Discrete event.** The above sample path consisted of only horizontal and vertical lines, as car arrivals and departures occurred at distinct points of time, what we refer to as events. Between two consecutive events, nothing happens – the graph is horizontal. When the number of events are finite, we call the simulation "discrete event."

In some systems the state changes all the time, not just at the time of some discrete events. For example, the water level in a reservoir with given in and outflows may change all the time. In such cases "continuous simulation" is more appropriate, although discrete event simulation can serve as an approximation.

Further consideration of discrete event simulations.

A.3.2.1.4 How is Simulation Performed?

Simulations may be performed manually. Most often, however, the system model is written either as a computer program or as some kind of input into simulator software.

System terminology:

- **State**: A variable characterizing an attribute in the system such as level of stock in inventory or number of jobs waiting for processing.
- **Event**: An occurrence at a point in time which may change the state of the system, such as arrival of a customer or start of work on a job.
- **Entity**: An object that passes through the system, such as cars in an intersection or orders in a factory. Often an event (e.g. arrival) is associated with an entity (e.g. customer).

- **Queue**: A queue is not only a physical queue of people, it can also be a task list, a buffer of finished goods waiting for transportation or any place where entities are waiting for something to happen for any reason.
- **Creating**: Creating is causing an arrival of a new entity to the system at some point in time.
- **Scheduling**: Scheduling is the act of assigning a new future event to an existing entity.
- **Random variable**: A random variable is a quantity that is uncertain, such as inter-arrival time between two incoming flights or number of defective parts in a shipment.
- **Random variate**: A random variate is an artificially generated random variable.
- **Distribution**: A distribution is the mathematical law which governs the probabilistic features of a random variable.

A.3.2.2 Development of Systems Simulation

Discrete event systems (DES) are dynamic systems which evolve in time by the occurrence of events at possibly irregular time intervals. DES abound in real-world applications. Examples include traffic systems, flexible manufacturing systems, computer-communications systems, production lines, coherent lifetime systems, and flow networks. Most of these systems can be modeled in terms of discrete events whose occurrence causes the system to change from one state to another. In designing, analyzing and operating such complex systems, one is interested not only in performance evaluation, but also in sensitivity analysis and optimization.

A typical stochastic system has a large number of control parameters that can have a significant impact on the performance of the system. To establish a basic knowledge of the behavior of a system under variation of input parameter values and to estimate the relative importance of the input parameters, sensitivity analysis applies small changes to the nominal values of input parameters. For systems simulation, variations of the input parameter values cannot be made infinitely small. The sensitivity of the performance measure with respect to an input parameter is therefore defined as (partial) derivative.

Sensitivity analysis is concerned with evaluating sensitivities (gradients, Hessian, etc.) of performance measures with respect to parameters of interest. It provides guidance for design and operational decisions and plays a pivotal role in identifying the most significant system parameters, as well as bottleneck subsystems. I have carried out research in the fields of sensitivity analysis and stochastic optimization of discrete event systems with an emphasis on computer simulation models. This part of lecture is dedicated to the estimation of an entire response surface of complex discrete event systems (DES) from a single sample path (simulation), such as the expected waiting time of a customer in a queuing network, with respect to the controllable parameters of the system, such as service rates, buffer sizes and routing probabilities. With the response surfaces at hand, we are able to perform sensitivity analysis and optimization of a DES from a single simulation, that is, to find the optimal parameters of the system and their sensitivities (derivatives), with respect to uncontrollable system parameters, such as arrival rates in a queuing network. We identified three distinct processes. Descriptive Analysis includes: Problem Identification and Formulation, Data Collection and Analysis, Computer Simulation Model Development, Validation, Verification and Calibration, and finally Performance Evaluation. Prescriptive Analysis: Optimization or Goal Seeking. These are necessary components for Post-prescriptive Analysis: Sensitivity, and What-If Analysis. The prescriptive simulation attempts to use simulation to prescribe decisions required to obtain specified results. It is subdivided into two topics- Goal Seeking and Optimization. Recent developments on “single-run” algorithms for the needed sensitivities (i.e. gradient, Hessian, etc.) make the prescriptive simulation feasible.

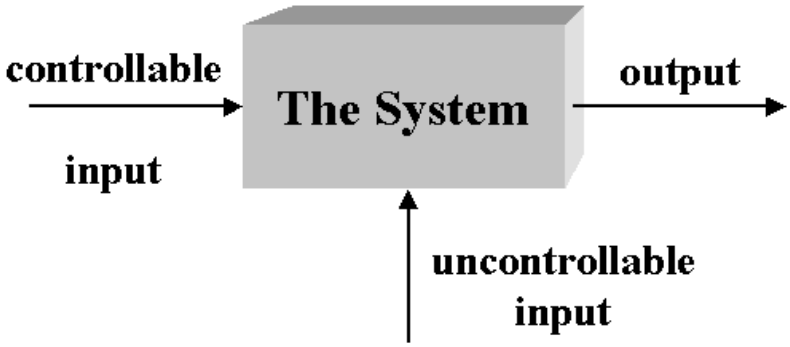

Figure A.7: Generic Design of DES.

A.3.2.2.1 Problem Formulation

Identify controllable and uncontrollable inputs. Identify constraints on the decision variables. Define measure of system performance and an objective function. Develop a preliminary model structure to interrelate the inputs and the measure of performance.

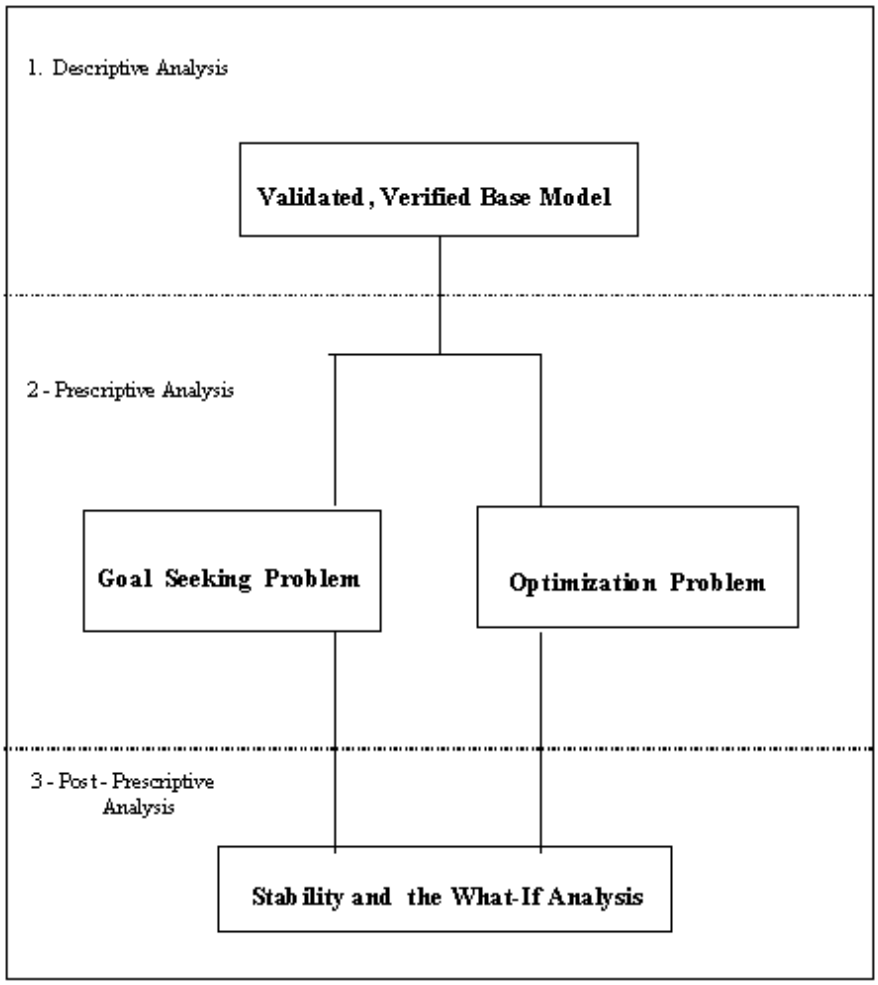

Figure A.8: A Development Process for Systems Simulation.

A.3.2.2.2 Data Collection and Analysis

Regardless of the method used to collect the data, the decision of how much to collect is a trade-off between cost and accuracy.

A.3.2.2.3 Simulation Model Development

Acquiring sufficient understanding of the system to develop an appropriate conceptual, logical and then simulation model is one of the most difficult tasks in simulation analysis.

A.3.2.2.4 Model Validation, Verification and Calibration

In general, verification focuses on the internal consistency of a model, while validation is concerned with the correspondence between the model and the reality. The term validation is applied to those processes which seek to determine whether or not a simulation is correct with respect to the "real" system. More prosaically, validation is concerned with the question "Are we building the right system?". Verification, on the other hand, seeks to answer the question "Are we building the system right?" Verification checks that the implementation of the simulation model (program) corresponds to the model. Validation checks that the model corresponds to reality. Calibration checks that the data generated by the simulation matches real (observed) data.

Validation: The process of comparing the model's output with the behavior of the phenomenon. In other words: comparing model execution to reality (physical or otherwise).

Verification: The process of comparing the computer code with the model to ensure that the code is a correct implementation of the model.

Calibration: The process of parameter estimation for a model. Calibration is a tweaking/tuning of existing parameters and usually does not involve the introduction of new ones, changing the model structure. In the context of optimization, calibration is an optimization procedure involved in system identification or during experimental design.

A.3.2.2.5 Input and Output Analysis

Discrete-event simulation models typically have stochastic components that mimic the probabilistic nature of the system under consideration. Successful input modeling requires a close match between the input model and the true underlying probabilistic mechanism associated with the system. The input data analysis is to model an element (e.g. arrival process, service times) in a discrete-event simulation given a data set collected on the element of interest. This stage performs intensive error checking on the input data, including external, policy, random and deterministic variables. System simulation experiment is to learn about its behavior. Careful planning, or designing, of simulation experiments is generally a great help, saving time and effort by providing efficient ways to estimate the effects of changes in the model's inputs on its outputs. Statistical experimental-design methods are mostly used in the context of simulation experiments.

Performance Evaluation and What-If Analysis: The `what-if' analysis is at the very heart of simulation models.

A.3.2.2.6 Sensitivity Estimation

Users must be provided with affordable techniques for sensitivity analysis if they are to understand which relationships are meaningful in complicated models.

A.3.2.2.7 Optimization

Traditional optimization techniques require gradient estimation. As with sensitivity analysis, the current approach for optimization requires intensive simulation to construct an approximate surface response function. Incorporating gradient estimation techniques into convergent algorithms such as Robbins-Monroe type algorithms for optimization purposes, will be considered.

A.3.2.2.8 Gradient Estimation Applications

There are a number of applications which measure sensitivity information, (i.e. the gradient, Hessian), Local information, Structural properties, Response surface generation, Goal-seeking problem, Optimization, What-if Problem, and Meta-modelling.

A.3.2.2.9 Report Generating

Report generation is a critical link in the communication process between the model and the end user.

A.3.2.3 Simulation Software Selection

The vast amount of simulation software available can be overwhelming for the new users. The following are only a random sample of software in the market today:

> ACSL, APROS, ARTIFEX, Arena, AutoMod, C++SIM, CSIM, Call$im, FluidFlow, GPSS, Gepasi, JavSim, MJX, MedModel, Mesquite, Multiverse, NETWORK, OPNET Modeler, POSES++, Simulat8, Powersim, QUEST, REAL, SHIFT, SIMPLE++, SIMSCRIPT, SLAM, SMPL, SimBank, SimPlusPlus, TIERRA, Witness, and javasim.

There are several things that make an ideal simulation package. Some are properties of the package, such as support, reactivity to bug notification, interface, etc. Some are properties of the user, such as their needs, their level of expertise, etc. For these reasons asking which package is best is a sudden failure of judgment. The first question to ask is for what purpose you need the software? Is it for education, teaching, student-projects or research?

The main question is: What are the important aspects to look for in a package? The answer depends on specific applications. However some general criteria are: Input facilities, Processing that allows some programming, Optimization capability, Output facilities, Environment including training and support services, Input-output statistical data analysis capability, and certainly the Cost factor.

You must know which features are appropriate for your situation, although, this is not based on a “Yes” or “No” judgment.

A.3.2.4 System Dynamics and Discrete Event Simulation

The modeling techniques used by system dynamics and discrete event simulations are often different at two levels: The modeler way of representing systems might be different, the underlying simulators’ algorithms are also different. Each technique is well tuned to the purpose it is intended. However, one may use a discrete event approach to do system dynamics and vice versa.

Traditionally, the most important distinction is the purpose of the modeling. The discrete event approach is to find, e.g. how many resources the decision maker needs such as how many trucks, and how to arrange the resources to avoid bottlenecks, i.e. excessive of waiting lines, waiting times, or inventories. While the system dynamics approach is to prescribe for the decision making to, e.g. timely respond to any changes, and how to change the physical structure, e.g. physical shipping delay time, so that inventories, sales, production, etc.

System dynamics is the rigorous study of problems in system behavior using the principles of feedback, dynamics and simulation. In more words system dynamics is characterized by:

- Searching for useful solutions to real problems, especially in social systems (businesses, schools, governments, ...) and the environment.
- Using computer simulation models to understand and improve such systems.
- Basing the simulation models on mental models, qualitative knowledge and numerical information.
- Using methods and insights from feedback control engineering and other scientific disciplines to assess and improve the quality of models.
- Seeking improved ways to translate scientific results into achieved implemented improvement.

Systems dynamics approach looks at systems at a very high level so is more suited to strategic analysis. Discrete event approach may look at subsystems for a detailed analysis and is more suited, e.g. to process re-engineering problems.

Systems dynamics is indicative, i.e. helps us understand the direction and magnitude of effects (i.e. where in the system do we need to make the changes), whereas discrete event approach is predictive (i.e. how many resources do we need to achieve a certain goal of throughout).

Systems dynamics analysis is continuous in time and it uses mostly deterministic analysis, whereas discrete event process deals with analysis in a specific time horizon and uses stochastic analysis.

Some interesting and useful areas of system dynamics modeling approach are:

- Short-term and long term forecasting of agricultural produce with special reference to field crops and perennial fruits such as grapes, which have significant processing sectors of different proportions of total output where both demand and supply side perspectives are being considered.
- Long term relationship between the financial statements of balance sheet, income statement and cash flow statement balanced against scenarios of the stock market's need to seek a stable/ growing share price combined with a satisfactory dividend and related return on shareholder funds policy.
- Managerial applications include the development and evaluation of short-term and long-term strategic plans, budget analysis and assessment, business audits and benchmarking.
- A modeler must consider both as complementary tools to each other. Systems dynamic to look at the high level problem and identify areas which need more detailed analysis. Then, use discrete event modeling tools to analyze (and predict) the specific areas of interest.

A.3.2.5 Parallel and Distributed Simulation

The increasing size of the systems and designs requires more efficient simulation strategies to accelerate the simulation process. Parallel and distributed simulation approaches seem to be a promising approach in this direction. Current topics under extensive research are:

- Synchronization, scheduling, memory management, randomized and reactive/adaptive algorithms, partitioning and load balancing.
- Synchronization in multi-user distributed simulation, virtual reality environments, HLA, and interoperability.
- System modeling for parallel simulation, specification, re-use of models/code, and parallelizing existing simulations.

- Language and implementation issues, models of parallel simulation, execution environments, and libraries.
- Theoretical and empirical studies, prediction and analysis, cost models, benchmarks, and comparative studies.
- Computer architectures, VLSI, telecommunication networks, manufacturing, dynamic systems, and biological/social systems.
- Web based distributed simulation such as multimedia and real time applications, fault tolerance, implementation issues, use of Java, and CORBA.

A.3.3 Simulation-Based Optimization Techniques

Discrete event simulation is the primary analysis tool for designing complex systems. Simulation, however, must be linked with a optimization techniques to be effectively used for systems design. We present several optimization techniques involving both continuous and discrete controllable input parameters subject to a variety of constraints. The aim is to determine the techniques most promising for a given simulation model.

Many man-made systems can be modeled as Discrete Event Systems (DES); examples are computer systems, communication networks, flexible manufacturing systems, production assembly lines, and traffic transportation systems. DES evolve with the occurrence of discrete events, such as the arrival of a job or the completion of a task, in contrast with continuously variable dynamic processes such as aerospace vehicles, which are primarily governed by differential equations. Owing to the complex dynamics resulting from stochastic interactions of such discrete events over time, the performance analysis and optimization of DES can be difficult tasks. At the same time, since such systems are becoming more widespread as a result of modern technological advances, it is important to have tools for analyzing and optimizing the parameters of these systems.

Analyzing complex DES often requires computer simulation. In these systems, the objective function may not be expressible as an explicit function of the input parameters; rather, it involves some performance measures of the system whose values can be found only by running the simulation model or by observing the actual system. On the other hand, due to the increasingly large size and inherent complexity of most man-made systems, purely analytical means are often insufficient for optimization. In these cases, one must resort to simulation, with its chief advantage being its generality, and its primary disadvantage being its cost in terms of time and money. Even though, in principle, some systems are analytically tractable, the analytical effort required to evaluate the solution may be so formidable that computer simulation becomes attractive. While the price for computing resources continue to dramatically decrease, one nevertheless can still obtain only a statistical estimate as opposed to an exact solution. For practical purposes, this is quite sufficient.

These man-made DES are costly, and therefore it is important to operate them as efficiently as possible. The high cost makes it necessary to find more efficient means of conducting simulation and optimizing its output. We consider optimizing an objective function with respect to a set of continuous and/or discrete controllable parameters subject to some constraints.

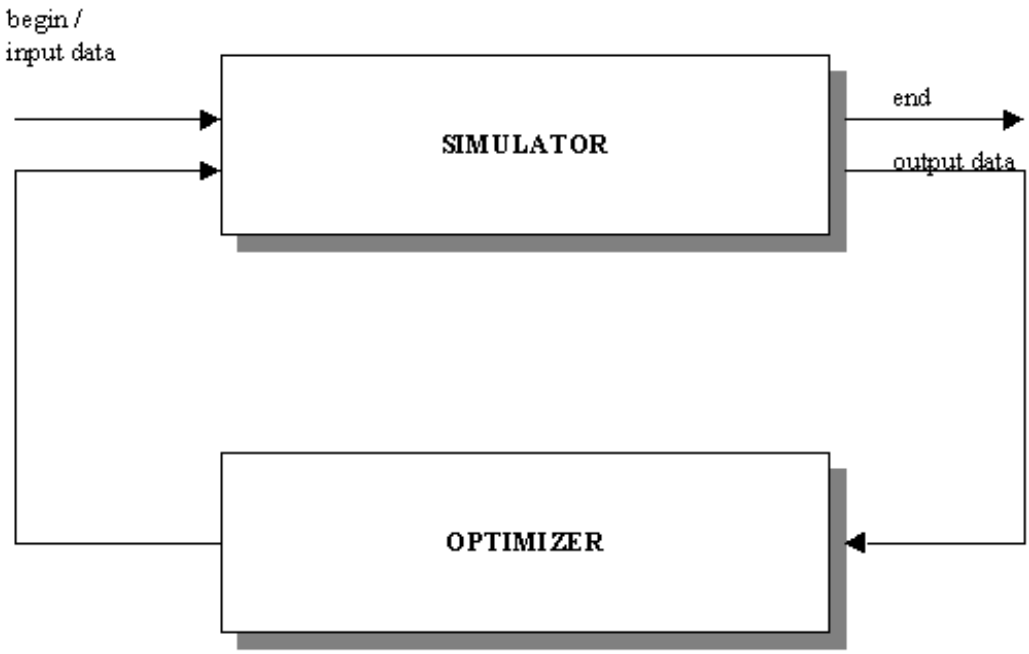

Figure A.9: Integration Scheme of Simulation and Optimization.

In almost all simulation models, an expected value can express the system's performance. Consider a system with continuous parameter v V, where V is the feasible region. Let

$$J(v) = E_{Y \mid v}[Z(Y)]$$

be the steady state expected performance measure, where Y is a random vector with known probability density function (pdf), f(y; v) depends on v, and Z is the performance measure.

In discrete event systems, Monte Carlo simulation is usually needed to estimate J(v) for a given value $v = v_0$. By the law of large numbers

$$J(v_0) = 1/n \, \Sigma \, Z(y_i)$$

converges to the true value, where y_i, i = 1, 2, ..., n are independent, identically distributed, random vector realizations of Y from $f(y; v_0)$, and n is the number of independent replications. The aim is to optimize J(v) with respect to v.

We shall group the optimization techniques for simulation into seven broad categories; namely, Deterministic Search, Pattern Search, Probabilistic Search, Evolutionary Techniques, Stochastic Approximation, Gradient Surface, and some Mixtures of the these techniques.

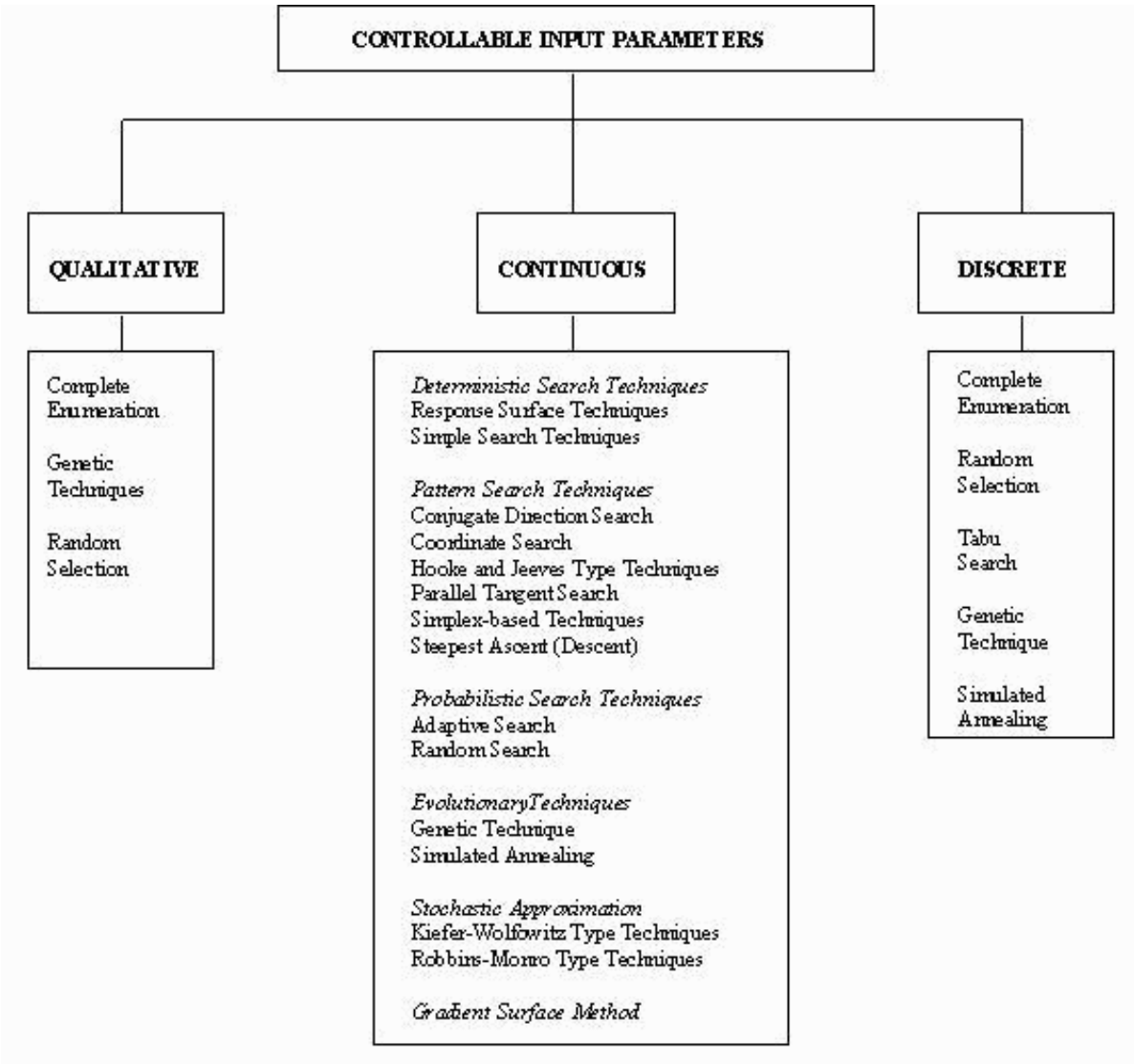

Figure A.10: A Classification of Optimization Techniques via Simulation.

A.3.3.1 Deterministic Search Techniques

A common characteristic of deterministic search techniques is that they are basically borrowed from deterministic optimization techniques. The deterministic objective function value required in the technique is now replaced with an estimate obtained from simulation. By having a reasonably accurate estimate, one hopes that the technique will perform well.

Deterministic search techniques include heuristic search, complete enumeration, and random search techniques.

A.3.3.1.1 Heuristic Search Technique

The heuristic search technique is probably most commonly used in optimizing response surfaces. It is also the least sophisticated scheme mathematically, and it can be thought of as an intuitive and experimental approach. The analyst determines the starting point and stopping rule based on previous experience with the system. After setting the input parameters (factors) to levels that appear reasonable, the analyst makes

a simulation run with the factors set at those levels and computes the value of the response function. If it appears to be a maximum (minimum) to the analyst, the experiment is stopped. Otherwise the analyst changes parameter settings and makes another run. This process continues until the analyst believes that the output has been optimized. Suffice it to say that, if the analyst is not intimately familiar with the process being simulated, this procedure can turn into a blind search and can expend an inordinate amount of time and computer resources without producing results commensurate with input. The heuristic search can be ineffective and inefficient in the hand of a novice.

A.3.3.1.2 Complete Enumeration and Random Techniques

The complete enumeration technique is not applicable to continuous cases, but in discrete space v it does yield the optimal value of the response variable. All factors (v) must assume a finite number of values for this technique to be applicable. Then, a complete factorial experiment is run. The analyst can attribute some degree of confidence to the determined optimal point when using this procedure. Although the complete enumeration technique yields the optimal point, it has a serious drawback. If the number of factors or levels per factor is large, the number of simulation runs required to find the optimal point can be exceedingly large. For example, suppose that an experiment is conducted with three factors having three, four, and five levels, respectively. Also suppose that five replications are desired to provide the proper degree of confidence. Then 300 runs of the simulator are required to find the optimal point. Hence, this technique should be used only when the number of unique treatment combinations is relatively small or a run takes little time.

The random search technique resembles the complete enumeration technique except that one selects a set of inputs at random. The simulated results based on the set that yields the maximum (minimum) value of the response function is taken to be the optimal point. This procedure reduces the number of simulation runs required to yield an ‘optimal’ result; however, there is no guarantee that the point found is actually the optimal point. Of course, the more points selected, the more likely the analyst is to achieve the true optimum. Note that the requirement that each factor assumes only a finite number of values is not a requirement in this scheme. Replications can be made on the treatment combinations selected, to increase the confidence in the optimal point. Which strategy is better, replicating a few points or looking at a single observation on more points, depends on the problem.

A.3.3.1.3 Response Surface Search

Response surface search attempts to fit a polynomial to J(v). If the design space v is suitably small, the performance function J(v) may be approximated by a response surface, typically a first order, or perhaps quadratic order in v, possibly after transformation, e.g. log (v). The response surface method (RSM) requires running the simulation in a first order experimental design to determine the path of steepest descent. Simulation runs made along this path continue, until one notes no improvement in J(v). The analyst then runs a new first order experimental design around the new ‘optimal’ point reached, and finds a new path of steepest descent. The process continues, until there is a lack of fit in the fitted first order surface. Then, one runs a second order design, and takes the optimum of the fittest second order surface as the estimated optimum.

Although it is desirable for search procedures to be efficient over a wide range of response surfaces, no current procedure can effectively overcome non-unimodality (surfaces having more than one local maximum or minimum). An obvious way to find the global optimal would be to evaluate all the local optima. One technique that is used when non-unimodality is known to exist, is called the “Las Vegas” technique. This search procedure estimates the distribution of the local optima by plotting the estimated J(v) for each local search against its corresponding search number. Those local searches that produce a response greater than any previous response are then identified and a curve is fitted to the data. This curve is then used to project the “estimated incremental” response that will be achieved by one more search.

The search continues until the value of the estimated improvement in the search is less than the cost of completing one additional search.

It should be noted that a well-designed experiment requires a sufficient number of replications so that the average response can be treated as a deterministic number for search comparisons. Otherwise, since replications are expensive, it becomes necessary to effectively utilize the number of simulation runs. Although each simulation is at a different setting of the controllable variables, one can use smoothing techniques such as exponential smoothing to reduce the required number of replications.

A.3.3.2 Pattern Search Techniques

Pattern search techniques assume that any successful set of moves used in searching for an approximated optimum is worth repeating. These techniques start with small steps; then, if these are successful, the step size increases. Alternatively, when a sequence of steps fails to improve the objective function, this indicates that shorter steps are appropriate so we may not overlook any promising direction. These techniques start by initially selecting a set of incremental values for each factor. Starting at an initial base point, they check if any incremental changes in the first variable yield an improvement. The resulting improved setting becomes the new intermediate base point. One repeats the process for each of the inputs until one obtains a new setting where the intermediate base points act as the initial base point for the first variable. The technique then moves to the new setting. This procedure is repeated, until further changes cannot be made with the given incremental values. Then, the incremental values are decreased, and the procedure is repeated from the beginning. When the incremental values reach a pre-specified tolerance, the procedure terminates; the most recent factor settings are reported as the solution.

A.3.3.2.1 Conjugate Direction Search

The conjugate direction search requires no derivative estimation, yet it finds the optimum of an N-dimensional quadratic surface after, at most, N-iterations, where the number of iterations is equal to the dimension of the quadratic surface. The procedure redefines the n dimensions so that a single variable search can be used successively. Single variable procedures can be used whenever dimensions can be treated independently. The optimization along each dimension leads to the optimization of the entire surface.

Two directions are defined to be conjugate whenever the cross-product terms are all zero. The conjugate direction technique tries to find a set of n dimensions that describes the surface such that each direction is conjugate to all others.

Using the above result, the technique attempts to find two search optima and replace the nth dimension of the quadratic surface by the direction specified by the two optimal points. Successively replacing the original dimension yields a new set of n dimensions in which, if the original surface is quadratic, all directions are conjugate to each other and appropriate for n single variable searches. While this search procedure appears to be very simple, we should point out that the selection of appropriate step sizes is most critical. The step size selection is more critical for this search technique because – during axis rotation – the step size does not remain invariant in all dimensions. As the rotation takes place, the best step size changes, and becomes difficult to estimate.

A.3.3.2.2 Steepest Ascent (Descent)

The steepest ascent (descent) technique uses a fundamental result from calculus (that the gradient points in the direction of the maximum increase of a function), to determine how the initial settings of the parameters should be changed to yield an optimal value of the response variable. The direction of movement is made proportional to the estimated sensitivity of the performance of each variable.

Although quadratic functions are sometimes used, one assumes that performance is linearly related to the change in the controllable variables for small changes. Assume that a good approximation is a linear form. The basis of the linear steepest ascent is that each controllable variable is changed in proportion to the magnitude of its slope. When each controllable variable is changed by a small amount, it is analogous to determining the gradient at a point. For a surface containing N controllable variables, this requires N points around the point of interest. When the problem is not an n-dimensional elliptical surface, the parallel-tangent points are extracted from bitangents and inflection points of occluding contours. Parallel tangent points are points on the occluding contour where the tangent is parallel to a given bitangent or the tangent at an inflection point.

A.3.3.2.3 Tabu Search Technique

An effective technique to overcome local optimality for discrete optimization is the Tabu Search technique. It explores the search space by moving from a solution to its best neighbor, even if this results in a deterioration of the performance measure value. This approach increases the likelihood of moving out of local optima. To avoid cycling, solutions that were recently examined are declared tabu (Taboo) for a certain number of iterations. Applying intensification procedures can accentuate the search in a promising region of the solution space. In contrast, diversification can be used to broaden the search to a less explored region. Much remains to be discovered about the range of problems for which the tabu search is best suited.

A.3.3.2.4 Hooke and Jeeves Type Techniques

The Hooke and Jeeves pattern search uses two kinds of moves; namely, an exploratory and a pattern move. The exploratory move is accomplished by doing a coordinate search in one pass through all the variables. This gives a new “base point” from which a pattern move is made. A pattern move is a jump in the pattern direction determined by subtracting the current base point from the previous base point. After the pattern move, another exploratory move is carried out at the point reached. If the estimate of J(v) is improved at the final point after the second exploratory move, it becomes the new base point. If it fails to show improvement, an exploratory move is carried out at the last base point with a smaller step in the coordinate search. The process stops when the step gets “small” enough.

A.3.3.2.5 Simplex-Based Techniques

The simplex-based technique performs simulation runs first at the vertices of the initial simplex; i.e. a polyhedron in the v-space having N+1 vertices. A subsequent simplex (moving towards the optimum) are formed by three operations performed on the current simplex: reflection, contraction, and expansion. At each stage of the search process, the point with the highest J(v) is replaced with a new point found via reflection through the centroid of the simplex. Depending on the value of J(v) at this new point, the simplex is either expanded, contracted, or unchanged. The simplex technique starts with a set of N+1 factor settings. These N+1 points are all the same distance from the current point. Moreover, the distance between any two points of these N+1 points is the same. Then, by comparing their response values, the technique eliminates the factor setting with the worst functional value and replaces it with a new factor setting, determined by the centroid of the N remaining factor settings and the eliminated factor setting. The resulting simplex either grows or shrinks, depending on the response value at the new factor settings. One repeats the procedure until no more improvement can be made by eliminating a point, and the resulting final simplex is small. While this technique will generally perform well for unconstrained problems, it may collapse to a point on a boundary of a feasible region, thereby causing the search to come to a premature halt. This technique is effective if the response surface is generally bowl-shaped even with some local optimal points.

A.3.3.3 Probabilistic Search Techniques

All probabilistic search techniques select trial points governed by a scan distribution, which is the main source of randomness. These search techniques include random search, pure adaptive techniques, simulated annealing, and genetic methods.

A.3.3.3.1 Random Search

A simple, but very popular approach is the random search, which centers a symmetric probability density function (pdf) [e.g. the normal distribution], about the current best location. The standard normal N(0, 1) is a popular choice, although the uniform distribution U[-1, 1] is also common.

A variation of the random search technique determines the maximum of the objective function by analyzing the distribution of J(v) in the bounded sub-region. In this variation, the random data are fitted to an asymptotic extreme-value distribution, and J* is estimated with a confidence statement. Unfortunately, these techniques cannot determine the location of J*, which can be as important as the J value itself. Some techniques calculate the mean value and the standard deviation of J(v) from the random data as they are collected. Assuming that J is distributed normally in the feasible region., the first trial, that yields a J-value two standard deviations within the mean value, is taken as a near-optimum solution.

A.3.3.3.2 Pure Adaptive Search

Various pure adaptive search techniques have been suggested for optimization in simulation. Essentially, these techniques move from the current solution to the next solution that is sampled uniformly from the set of all better feasible solutions.

A.3.3.4 Evolutionary Techniques

Nature is a robust optimizer. By analyzing nature's optimization mechanism we may find acceptable solution techniques to intractable problems. Two concepts that have most promise are simulated annealing and the genetic techniques.

A.3.3.4.1 Simulated Annealing

Simulated annealing (SA) borrows its basic ideas from statistical mechanics. A metal cools, and the electrons align themselves in an optimal pattern for the transfer of energy. In general, a slowly cooling system, left to itself, eventually finds the arrangement of atoms, which has the lowest energy. The is the behavior, which motivates the method of optimization by SA. In SA we construct a model of a system and slowly decrease the "temperature" of this theoretical system, until the system assumes a minimal energy structure. The problem is how to map our particular problem to such an optimizing scheme.

SA as an optimization technique was first introduced to solve problems in discrete optimization, mainly combinatorial optimization. Subsequently, this technique has been successfully applied to solve optimization problems over the space of continuous decision variables. SA is a simulation optimization technique that allows random ascent moves in order to escape the local minima, but a price is paid in terms of a large increase in the computational time required. It can be proven that the technique will find an approximated optimum. The annealing schedule might require a long time to reach a true optimum.

A.3.3.4.2 Genetic Techniques

Genetic techniques (GT) are optimizers that use the ideas of evolution to optimize a system that is too difficult for traditional optimization techniques. Organisms are known to optimize themselves to adapt to their environment.

GT differ from traditional optimization procedures in that GT work with a coding of the decision parameter set, not the parameters themselves; GT search a population of points, not a single point; GT use objective function information, not derivatives or other auxiliary knowledge; and finally, GT use probabilistic transition rules, not deterministic rules. GT are probabilistic search optimizing techniques that do not require mathematical knowledge of the response surface of the system, which they are optimizing. They borrow the paradigms of genetic evolution, specifically selection, crossover, and mutation.

Selection: The current points in the space are ranked in terms of their fitness by their respective response values. A probability is assigned to each point that is proportional to its fitness, and parents (a mating pair) are randomly selected.

Crossover: The new point, or offspring, is chosen, based on some combination of the genetics of the two parents.

Mutation: The location of offspring is also susceptible to mutation, a process, which occurs with probability p, by which a offspring is replaced randomly by a new offspring location.

A generalized GT generates p new offspring at once and kills off all of the parents. This modification is important in the simulation environment. GT are well suited for qualitative or policy decision optimization such as selecting the best queuing disciplines or network topologies. They can be used to help determine the design of the system and its operation. For applications of GT to inventory systems, job-shop, and computer time-sharing problems. GT do not have certain shortcomings of other optimization techniques, and they will usually result in better calculated optima than those found with the traditionally techniques. They can search a response surface with many local optima and find (with a high probability) the approximate global optimum. One may use GT to find an area of potential interest, and then resort to other techniques to find the optimum. Recently, several classical GT principles have been challenged.

Differential Evolution: Differential Evolution (DE) is a genetic type of algorithm for solving continuous stochastic function optimization. The basic idea is to use vector differences for perturbing the vector population. DE adds the weighted difference between two population vectors to a third vector. This way, no separate probability distribution has to be used, which makes the scheme completely self-organizing.

A.3.3.4.3 A Short Comparison

When performing search techniques in general, and simulated annealing or genetic techniques specifically, the question of how to generate the initial solution arises. Should it be based on a heuristic rule or on a randomly generated one? Theoretically, it should not matter, but in practice this may depend on the problem. In some cases, a pure random solution systematically produces better final results. On the other hand, a good initial solution may lead to lower overall run times. This can be important, for example, in cases where each iteration takes a relatively long time; therefore, one has to use some clever termination rule. Simulation time is a crucial bottleneck in an optimization process. In many cases, a simulation is run several times with different initial solutions. Such a technique is most robust, but it requires the maximum number of replications compared with all other techniques. The pattern search technique applied to small problems with no constraints or qualitative input parameters requires fewer replications than the GT. GT, however, can easily handle constraints, and have lower computational complexity. Finally, simulated annealing can be embedded within the Tabu search to construct a probabilistic technique for global optimization.

A.3.3.5 Stochastic Approximation Techniques

Two related stochastic approximation techniques have been proposed, one by Robbins and Monro and one by Kiefer and Wolfowitz. The first technique was not useful for optimization until an unbiased estimator

for the gradient was found. Kiefer and Wolfowitz developed a procedure for optimization using finite differences. Both techniques are useful in the optimization of noisy functions, but they did not receive much attention in the simulation field until recently. Generalization and refinement of stochastic approximation procedures give rise to a weighted average, and stochastic quasi-gradient methods. These deal with constraints, non-differentiable functions, and some classes of non-convex functions, among other things.

A.3.3.5.1 Kiefer-Wolfowitz Type Techniques

Kiefer and Wolfowitz proposed a finite difference approximation to the derivative. One version of the Kiefer-Wolfwitz technique uses two-sided finite differences. The first fact to notice about the K-W estimate is that it requires 2N simulation runs, where N is the dimension of vector parameter v. If the decision maker is interested in gradient estimation with respect to each of the components of v, then 2N simulations must be run for each component of v . This is inefficient. The second fact is that it may have a very poor variance, and it may result in numerical calculation difficulties.

A.3.3.5.2 Robbins-Monro Type Techniques

The original Robbins-Monro (R-M) technique is not an optimization scheme, but rather a root finding procedure for functions whose exact values are not known but are observed with noise. Its application to optimization is immediate: use the procedure to find the root of the gradient of the objective function.

Interest was renewed in the R-M technique as a means of optimization, with the development of the perturbation analysis, score function (known also as likelihood ratio method), and frequency domain estimates of derivatives. Optimization for simulated systems based on the R-M technique is known as a "single-run" technique. These procedures optimize a simulation model in a single run simulation with a run length comparable to that required for a single iteration step in the other methods. This is achieved essentially be observing the sample values of the objective function and, based on these observations, updating the values of the controllable parameters while the simulation is running, that is, without restarting the simulation. This observing-updating sequence is done repeatedly, leading to an estimate of the optimum at the end of a single-run simulation. Besides having the potential of large computational savings, this technique can be a powerful tool in real-time optimization and control, where observations are taken as the system is evolving in time.

A.3.3.6 Gradient Surface Method

One may combine the gradient-based techniques with the response surface methods (RSM) for optimization purposes. One constructs a response surface with the aid of n response points and the components of their gradients.

The gradient surface method (GSM) combines the virtue of RSM with that of the single-run, gradient estimation techniques such as Perturbation Analysis, and Score Function techniques. A single simulation experiment with little extra work yields N + 1 pieces of information; i.e. one response point and N components of the gradient. This is in contrast to crude simulation, where only one piece of information, the response value, is obtained per experiment. Thus by taking advantage of the computational efficiency of single-run gradient estimators. In general, N-fold fewer experiments will be needed to fit a global surface compared to the RSM. At each step, instead of using Robbins-Monro techniques to locate the next point locally, we determine a candidate for the next point globally, based on the current global fit to the performance surface.

The GSM approach has the following advantages; The technique can quickly get to the vicinity of the optimal solution because its orientation is global [23, 39]. Thus, it produces satisfying solutions quickly;

Like RSM, it uses all accumulated information; And, in addition, it uses gradient surface fitting, rather than direct performance response-surface fitting via single-run gradient estimators. This significantly reduces the computational efforts compared with RSM. Similar to RSM, GSM is less sensitive to estimation error and local optimality; and, finally, it is an on-line technique, the technique may be implemented while the system is running.

A typical optimization scheme involves two phases: a Search Phase and an Iteration Phase. Most results in analytic computational complexity assume that good initial approximations are available, and deal with the iteration phase only. If enough time is spent in the initial search phase, we can reduce the time needed in the iteration phase. The literature contains papers giving conditions for the convergence of a process; a process has to be more than convergent in order to be computationally interesting. It is essential that we be able to limit the cost of computation. In this sense, GSM can be thought of as helping the search phase and as an aid to limit the cost of computation. One can adopt standard or simple devices for issues such as stopping rules.

For on-line optimization, one may use a new design in GSM called 'single direction' design. Since for on-line optimization it may not be advisable or feasible to disturb the system, random design usually is not suitable.

A.3.3.7 Post-Solution Analysis

Stochastic models typically depend upon various uncertain and uncontrollable input parameters that must be estimated from existing data sets. We focus on the statistical question of how input-parameter uncertainty propagates through the model into output- parameter uncertainty. The sequential stages are descriptive, prescriptive and post-prescriptive analysis.

A.3.3.8 Rare Event Simulation

Large deviations can be used to estimate the probability of rare events, such as buffer overflow, in queuing networks. It is simple enough to be applied to very general traffic models, and sophisticated enough to give insight into complex behavior.

Simulation has numerous advantages over other approaches to performance and dependability evaluation; most notably, its modelling power and flexibility. For some models, however, a potential problem is the excessive simulation effort (time) required to achieve the desired accuracy. In particular, simulation of models involving rare events, such as those used for the evaluation of communications and highly-dependable systems, is often not feasible using standard techniques. In recent years, there have been significant theoretical and practical advances towards the development of efficient simulation techniques for the evaluation of these systems.

Methodologies include: Techniques based on importance sampling, The "restart" method, and Hybrid analytic/simulation techniques among newly devised approaches.

A.3.3.9 Conclusion

With the growing incidence of computer modeling and simulation, the scope of simulation domain must be extended to include much more than traditional optimization techniques. Optimization techniques for simulation must also account specifically for the randomness inherent in estimating the performance measure and satisfying the constraints of stochastic systems. We described the most widely used optimization techniques that can be effectively integrated with a simulation model. We also described techniques for post-solution analysis with the aim of theoretical unification of the existing techniques. All techniques were presented in step-by-step format to facilitate implementation in a variety of operating systems and computers, thus improving portability.

General comparisons among different techniques in terms of bias, variance, and computational complexity are not possible. However, a few studies rely on real computer simulations to compare different techniques in terms of accuracy and number of iterations. Total computational effort for reduction in both the bias and variance of the estimate depends on the computational budget allocated for a simulation optimization. No single technique works effectively and/or efficiently in all cases.

The simplest technique is the random selection of some points in the search region for estimating the performance measure. In this technique, one usually fixes the number of simulation runs and takes the smallest (or largest) estimated performance measure as the optimum. This technique is useful in combination with other techniques to create a multi-start technique for global optimization. The most effective technique to overcome local optimality for discrete optimization is the Tabu Search technique. In general, the probabilistic search techniques, as a class, offer several advantages over other optimization techniques based on gradients. In the random search technique, the objective function can be non-smooth or even have discontinuities. The search program is simple to implement on a computer, and it often shows good convergence characteristics in noisy environments. More importantly, it can offer the global solution in a multi-modal problem, if the technique is employed in the global sense. Convergence proofs under various conditions are given in.

The Hooke-Jeeves search technique works well for unconstrained problems with less than 20 variables; pattern search techniques are more effective for constrained problems. Genetic techniques are most robust and can produce near-best solutions for larger problems. The pattern search technique is most suitable for small size problems with no constraint, and it requires fewer iterations than the genetic techniques. The most promising techniques are the stochastic approximation, simultaneous perturbation, and the gradient surface methods. Stochastic approximation techniques using perturbation analysis, score function, or simultaneous perturbation gradient estimators, optimize a simulation model in a single simulation run. They do so by observing the sample values of the objective function, and based on these observations, the stochastic approximation techniques update the values of the controllable parameters while the simulation is running and without restarting the simulation. This observing-updating sequence, done repeatedly, leads to an estimate of the optimum at the end of a single-run simulation. Besides having the potential of large savings in computational effort in the simulation environment, this technique can be a powerful tool in real-time optimization and control, where observations are taken as the system is evolving over time.

Response surface methods have a slow convergence rate, which makes them expensive. The gradient surface method combines the advantages of the response surface methods (RSM) and efficiency of the gradient estimation techniques, such as infinitesimal perturbation analysis, score function, simultaneous perturbation analysis, and frequency domain technique. In the gradient surface method (GSM) the gradient is estimated, and the performance gradient surface is estimated from observations at various points, similar to the RSM. Zero points of the successively approximating gradient surface are then taken as the estimates of the optimal solution. GSM is characterized by several attractive features: it is a single run technique and more efficient than RSM; at each iteration step, it uses the information from all of the data points rather than just the local gradient; it tries to capture the global features of the gradient surface and thereby quickly arrive in the vicinity of the optimal solution, but close to the optimum, they take many iterations to converge to stationary points. Search techniques are therefore more suitable as a second phase. The main interest is to figure out how to allocate the total available computational budget across the successive iterations.

For when the decision variable is qualitative, such as finding the best system configuration, a random or permutation test is proposed. This technique starts with the selection of an appropriate test statistic, such as the absolute difference between the mean responses under two scenarios. The test value is computed for the original data set. The data are shuffled (using a different seed); the test statistic is computed for the shuffled data; and the value is compared to the value of the test statistic for the original, un-shuffled data.

If the statistics for the shuffled data are greater than or equal to the actual statistic for the original data, then a counter c, is incremented by 1. The process is repeated for any desired m number of times. The final step is to compute (c+1)/(m+1), which is the significant level of the test. The null hypothesis is rejected if this significance level is less than or equal to the specified rejection level for the test. There are several important aspects to this non-parametric test. First, it enables the user to select the statistic. Second, assumptions such as normality or equality of variances made for the t-test, ranking-and-selection, and multiple-comparison procedures, are no longer needed. A generalization is the well-known bootstrap technique.

- Computational studies of techniques for systems with a large number of controllable parameters and constraints.
- Effective combinations of several efficient techniques to achieve the best results under constraints on computational resources.
- Development of parallel and distributed schemes and an expert system that incorporates all available techniques.

A.3.4 Simulation Products

A.3.4.1 Crystal Ball 2000 Professional Edition

Crystal Ball Pro is a suite of products that includes Crystal Ball 2000 Standard Edition (Monte Carlo simulation software for spreadsheet risk analysis), OptQuest (global optimization software that finds the best solutions for spreadsheets under conditions of uncertainty), CB Predictor (time-series forecasting software for historic data), Crystal Ball and CB Predictor Developer Kits (automate and customize your Crystal Ball simulations) and Crystal Ball Tools (improve your ability to analyze variables and save on the time spent building models).

OS: Windows 95, Windows 98, Windows NT; Microsoft Office 95 (or later)

Vendor: Decisioneering Inc. http://www.decisioneering.com

A.3.4.2 DecisionPro 3.0

DecisionPro picks up where your spreadsheet leaves off. It helps you make the best possible business decisions by applying proven management techniques such as decision tree analysis, Monte Carlo simulation, linear programming, advanced forecasting methods, and others. By integrating all of the key quantitative methods in management into a single application, DecisionPro gives you unprecedented power and flexibility. What's more, Decision Pro's innovative interface makes this power so easy to apply that you will find yourself using it even for routine problems.

OS: Windows 95, Windows 98, Windows NT

Vendor: Vanguard Software Corporation http://www.vanguardsw.com

Demo: http://www.vanguardsw.com/models/download.dsb?1

A.3.4.3 Deneb/QUEST

The Queuing Event Simulation Tool, Deneb/QUEST represents a quantum leap in the modeling and analysis of manufacturing systems. Detailed physical system properties combined with interactive 3D graphics and visual analysis deliver a new level of ease, power, and accuracy. Deneb/QUEST gives new meaning to ``interactive modeling'' and ``what-if' evaluations. The effects of any model change can

be instantly seen just by pressing the RUN button. True 3D animation occurs in real-time, not as a recording. Lengthy edit/compile/run/analyze cycles are eliminated. Now, efficiently explore production scenarios, product mixes, and failure responses for machine and labor utilization, bottleneck and throughput evaluation. Analyze and justify production costs, capital investments, and monitor the value of work-in-process, inventory and ROI.

OS: Windows 95, Windows 98, Windows NT

Vendor: Deneb Robotnics http://www.deneb.com

A.4 DATA ANALYSIS AND MINING

A.4.1 Data Mining and Analysis Presentation

A.4.1.1 Data Mining Overview

Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Most companies already collect and refine massive quantities of data. Data mining techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources, and can be integrated with new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing computers, data mining tools can analyze massive databases to deliver answers to questions such as, “Which clients are most likely to respond to my next promotional mailing, and why?”

This paper provides an introduction to the basic technologies of data mining. Examples of profitable applications illustrate its relevance to today’s business environment as well as a basic description of how data warehouse architectures can evolve to deliver the value of data mining to end users.

A.4.1.2 Foundations of Data Mining

Data mining techniques are the result of a long process of research and product development. This evolution began when business data was first stored on computers, continued with improvements in data access, and more recently, generated technologies that allow users to navigate through their data in real time. Data mining takes this evolutionary process beyond retrospective data access and navigation to prospective and proactive information delivery. Data mining is ready for application in the business community because it is supported by three technologies that are now sufficiently mature: Massive data collection, Powerful multiprocessor computers, Data mining algorithms.

Data mining algorithms embody techniques that have existed for at least 10 years, but have only recently been implemented as mature, reliable, understandable tools that consistently outperform older statistical methods.

In the evolution from business data to business information, each new step has built upon the previous one. For example, dynamic data access is critical for drill-through in data navigation applications, and the

ability to store large databases is critical to data mining. From the user's point of view, the four steps listed in the table were revolutionary because they allowed new business questions to be answered accurately and quickly.

Table A.3: Steps in the Evolution of Data Mining

Evolutionary Step	Business Question	Enabling Technologies	Product Providers	Characteristics
Data Collection (1960s)	"What was my total revenue in the last five years?"	Computers, tapes, disks	IBM, CDC	Retrospective, static data delivery
Data Access (1980s)	"What were unit sales in New England last March?"	Relational databases (RDBMS), Structured Query Language (SQL), ODBC	Oracle, Sybase, Informix, IBM, Microsoft	Retrospective, dynamic data delivery at record level
Data Warehousing and Decision Support (1990s)	"What were unit sales in New England last March? Drill down to Boston."	On-line analytic processing (OLAP), multidimensional databases, data warehouses	Pilot, Comshare, Arbor, Cognos, Microstrategy	Retrospective, dynamic data delivery at multiple levels
Data Mining (Emerging Today)	"What's likely to happen to Boston unit sales next month? Why?"	Advanced algorithms, multiprocessor computers, massive databases	Pilot, Lockheed, IBM, SGI, numerous startups (nascent industry)	Prospective, proactive information delivery

The core components of data mining technology have been under development for decades, in research areas such as statistics, artificial intelligence, and machine learning. Today, the maturity of these techniques, coupled with high-performance relational database engines and broad data integration efforts, make these technologies practical for current data warehouse environments.

A.4.1.3 The Scope of Data Mining

Data mining derives its name from the similarities between searching for valuable business information in a large database – for example, finding linked products in gigabytes of store scanner data – and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

- Automated prediction of trends and behaviors. Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data – quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.
- Automated discovery of previously unknown patterns. Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the

analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

- Data mining techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When data mining tools are implemented on high performance parallel processing systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions.

Databases can be larger in both depth and breadth:

- More columns. Analysts must often limit the number of variables they examine when doing hands-on analysis due to time constraints. Yet variables that are discarded because they seem unimportant may carry information about unknown patterns. High performance data mining allows users to explore the full depth of a database, without pre-selecting a subset of variables.
- More rows. Larger samples yield lower estimation errors and variance, and allow users to make inferences about small but important segments of a population.

The most commonly used techniques in data mining are:

- Artificial neural networks: Non-linear predictive models that learn through training and resemble biological neural networks in structure.
- Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID).
- Genetic algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution.
- Nearest neighbor method: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset (where k is greater than or equal to 1). Sometimes called the k-nearest neighbor technique.
- Rule induction: The extraction of useful if-then rules from data based on statistical significance.

Many of these technologies have been in use for more than a decade in specialized analysis tools that work with relatively small volumes of data. These capabilities are now evolving to integrate directly with industry-standard data warehouse and OLAP platforms.

A.4.1.4 An Architecture for Data Mining

To best apply these advanced techniques, they must be fully integrated with a data warehouse as well as flexible interactive business analysis tools. Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on. This figure illustrates an architecture for advanced analysis in a large data warehouse.

A.4.1.4.1 Integrated Data Mining Architecture

The ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on, and should be optimized for flexible and fast data access.

An OLAP (On-Line Analytical Processing) server enables a more sophisticated end-user business model to be applied when navigating the data warehouse. The multidimensional structures allow the user to analyze the data as they want to view their business – summarizing by product line, region, and other key perspectives of their business. The Data Mining Server must be integrated with the data warehouse and the OLAP server to embed ROI-focused business analysis directly into this infrastructure. An advanced, process-centric metadata template defines the data mining objectives for specific business issues like campaign management, prospecting, and promotion optimization. Integration with the data warehouse enables operational decisions to be directly implemented and tracked. As the warehouse grows with new decisions and results, the organization can continually mine the best practices and apply them to future decisions.

This design represents a fundamental shift from conventional decision support systems. Rather than simply delivering data to the end user through query and reporting software, the Advanced Analysis Server applies users' business models directly to the warehouse and returns a proactive analysis of the most relevant information. These results enhance the metadata in the OLAP Server by providing a dynamic metadata layer that represents a distilled view of the data. Reporting, visualization, and other analysis tools can then be applied to plan future actions and confirm the impact of those plans.

A.4.1.5 Data Mining Applications

A wide range of companies have deployed successful applications of data mining. While early adopters of this technology have tended to be in information-intensive industries such as financial services and direct mail marketing, the technology is applicable to any company looking to leverage a large data warehouse to better manage their customer relationships. Two critical factors for success with data mining are: a large, well-integrated data warehouse and a well-defined understanding of the business process within which data mining is to be applied (such as customer prospecting, retention, campaign management, and so on).

Some successful application areas include:

- A pharmaceutical company can analyze its recent sales force activity and their results to improve targeting of high-value physicians and determine which marketing activities will have the greatest impact in the next few months. The data needs to include competitor market activity as well as information about the local health care systems. The results can be distributed to the sales force via a wide-area network that enables the representatives to review the recommendations from the perspective of the key attributes in the decision process. The ongoing, dynamic analysis of the data warehouse allows best practices from throughout the organization to be applied in specific sales situations.
- A credit card company can leverage its vast warehouse of customer transaction data to identify customers most likely to be interested in a new credit product. Using a small test mailing, the attributes of customers with an affinity for the product can be identified. Recent projects have indicated more than a 20-fold decrease in costs for targeted mailing campaigns over conventional approaches.
- A diversified transportation company with a large direct sales force can apply data mining to identify the best prospects for its services. Using data mining to analyze its own customer

experience, this company can build a unique segmentation identifying the attributes of high-value prospects. Applying this segmentation to a general business database such as those provided by Dun & Bradstreet can yield a prioritized list of prospects by region.

- A large consumer package goods company can apply data mining to improve its sales process to retailers. Data from consumer panels, shipments, and competitor activity can be applied to understand the reasons for brand and store switching. Through this analysis, the manufacturer can select promotional strategies that best reach their target customer segments.

Each of these examples have a clear common ground. They leverage the knowledge about customers implicit in a data warehouse to reduce costs and improve the value of customer relationships. These organizations can now focus their efforts on the most important (profitable) customers and prospects, and design targeted marketing strategies to best reach them.

A.4.1.6 Conclusion

Comprehensive data warehouses that integrate operational data with customer, supplier, and market information have resulted in an explosion of information. Competition requires timely and sophisticated analysis on an integrated view of the data. However, there is a growing gap between more powerful storage and retrieval systems and the users' ability to effectively analyze and act on the information they contain. Both relational and OLAP technologies have tremendous capabilities for navigating massive data warehouses, but brute force navigation of data is not enough. A new technological leap is needed to structure and prioritize information for specific end-user problems. The data mining tools can make this leap. Quantifiable business benefits have been proven through the integration of data mining with current information systems, and new products are on the horizon that will bring this integration to an even wider audience of users.

A.4.2 Data Mining Products

The Product Features table includes many of the mainstream data mining products, focusing on those with which Exclusive Ore has had hands-on experience. While we attempt to keep the table as current as possible, we encourage you to use the hot-links to the vendor sites for the latest information on features and functionality.

Key: NN = Neural Net; Tree = Decision Tree; Naïve Bayes; k-Mns = K Means; k-NN = k-Nearest Neighbor; Stats = Traditional Statistical Techniques; Pred = Prediction; Time Series; Clust = Clustering; Assoc = Association; Win 32 = Windows 95/98/NT; UNIX; Par = Parallel Scalability (in at least one OS); API SDK = API or SDK available; SQL Ext = Uses Special SQL Extensions

Table A.4: Data Mining Products

Company	Product	NN	Tree	Naïve Bayes	k-Mns	k-NN	Stats	Pred	Time Series	Clust	Assoc	Win 32	UNIX	Par	API SDK	SQL Ext
Angoss International Ltd.	KnowledgeSEEKER		Yes					Yes				Yes	Yes	Yes		
	KnowledgeSTUDIO	Yes	Yes		Yes			Yes		Yes		Yes	Yes	Yes	Yes	Yes
Business Objects	BusinessMiner		Yes									Yes				
Cognos Incorporated	4Thought	Yes						Yes	Yes			Yes				
Informix Software Inc.	Red Brick Data Mine			Yes				Yes				Yes	Yes			Yes
International Business Machines	Intelligent Miner	Yes	Yes			Yes		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Accrue Software	Decision Series	Yes	Yes	Yes				Yes		Yes	Yes		Yes	Yes	Yes	
NeuralWare	NeuralSIM	Yes						Yes				Yes				
Oracle Corp.	Darwin	Yes	Yes			Yes		Yes					Yes	Yes		
RightPoint Software	DataCruncher			Yes				Yes				Yes	Yes			
Salford Systems	CART		Yes					Yes				Yes	Yes			
SAS Institute	Enterprise Miner	Yes	Yes				Yes	Yes	Yes	Yes	Yes	Yes	Yes			
SGI	MineSet		Yes	Yes	Yes			Yes		Yes	Yes	Yes	Yes	Yes	Yes	
SPSS, Inc.	Answer Tree		Yes				Yes	Yes				Yes	Yes			
	Clementine	Yes	Yes					Yes	Yes	Yes	Yes	Yes	Yes			
	Neural Connection	Yes					Yes	Yes				Yes	Yes			
Tandem Computers Incorporated	Object-Relational Technology												Yes	Yes		Yes
Unica Technology	Pattern Recognition Workbench	Yes			Yes	Yes	Yes	Yes	Yes	Yes		Yes			Yes	
	Model 1	Yes	Yes	Yes	Yes		Yes	Yes				Yes	Yes	Yes		

A.5 ARTIFICIAL INTELLIGENCE

A.5.1 Artificial Intelligence Presentation

A.5.1.1 What is Artificial Intelligence (AI)?

Artificial intelligence can be viewed from a variety of perspectives. From the perspective of **intelligence** *artificial intelligence* is making machines "intelligent" – acting as we would expect people to act. The inability to distinguish computer responses from human responses is called the Turing test. Intelligence requires knowledge, and expert problem solving restricts domain to allow including significant relevant knowledge. From a **research** perspective *artificial intelligence* is the study of how to make computers do things which, at the moment, people do better.

AI began in the early 1960s, the first attempts were game playing (checkers), theorem proving (a few simple theorems) and general problem solving (only very simple tasks). General problem solving was much more difficult than originally anticipated. Researchers were unable to tackle problems routinely handled by human experts. The name "artificial intelligence" came from the roots of the area of study. AI researchers are active in a variety of **domains**.

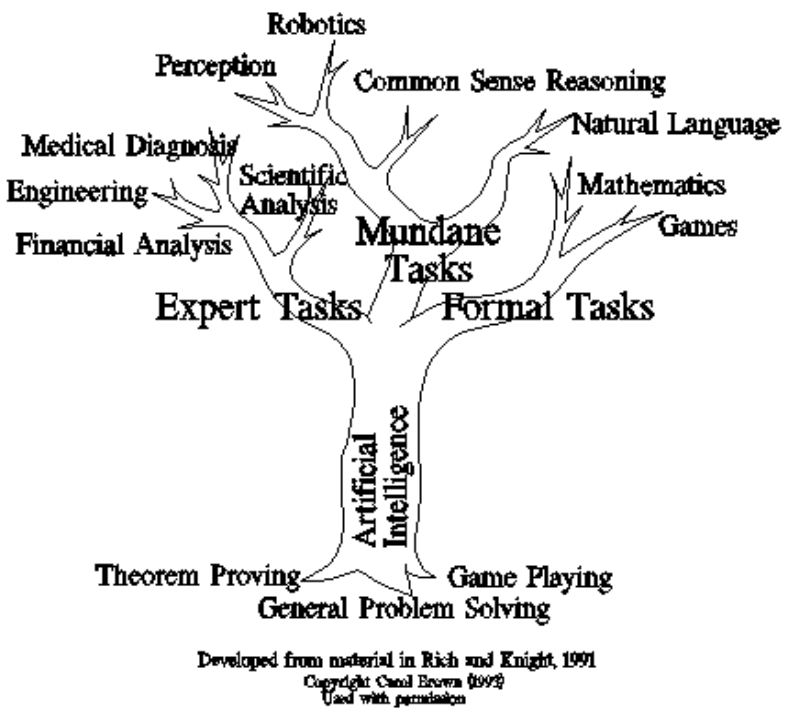

Figure A.11: Task Domains of Artificial Intelligence.

Artificial Intelligence domains:

- Formal Tasks (mathematics, games); and
- Expert tasks (financial analysis, medical diagnostics, engineering, scientific analysis, and other areas).

From a **business** perspective AI is a set of very powerful tools, and methodologies for using those tools to solve business problems.

From a **programming** perspective, AI includes the study of symbolic programming, problem solving, and search.

Typically AI programs focus on symbols rather than numeric processing.

- Problem solving – achieve goals.
- Search – seldom access a solution directly. Search may include a variety of techniques.
- AI programming languages include:
  - **LISP**, developed in the 1950s, is the early programming language strongly associated with AI. LISP is a functional programming language with procedural extensions. LISP (LISt Processor) was specifically designed for processing heterogeneous lists, typically a list of symbols. Features of LISP that made it attractive to AI researchers included run-time type checking, higher order functions (functions that have other functions as parameters), automatic memory management (garbage collection) and an interactive environment.
  - The second language strongly associated with AI is PROLOG. PROLOG was developed in the 1970s. **PROLOG** is based on first order logic. PROLOG is declarative in nature and has facilities for explicitly limiting the search space.

Object-oriented languages are a class of languages more recently used for AI programming. Important features of object-oriented languages include:

- Concepts of objects and messages;
- Objects bundle data and methods for manipulating the data;
- Sender specifies what is to be done receiver decides how to do it; and
- Inheritance (object hierarchy where objects inherit the attributes of the more general class of objects).

Examples of object-oriented languages are Smalltalk, Objective C, C++, JAVA. Object oriented extensions to LISP (CLOS – Common LISP Object System) and PROLOG (L&O – Logic & Objects) are also used.

A.5.1.2 AI Tools and Languages

AI researchers have often found that existing software tools and programming languages, and standard design and development methodologies, were too restrictive, and therefore developed new special purpose versions. However many of the ideas have been taken up more generally, so that the distinctions are no longer clear.

Some of the reasons for the special requirements are:

- Many AI problems are not well defined initially: in such cases the process of developing and testing software helps to clarify the problem.
- Often the tasks require development of very complex designs which are hard to get right. This requires powerful interactive testing facilities.
- AI problems often require different kinds of knowledge to be represented and different kinds of inference mechanisms to be used.
- Intelligent systems may need to modify themselves at run time.
- AI systems often need to construct complex networks of temporary structures, where different portions have different life-spans.

These requirements have led to the development of languages which:

- Are extendable using macros and other facilities.
- Include support for automatic garbage collection (once described by a well known computer scientist as a luxury for the idle rich).
- Use interpreters or incremental compilers.
- Have built in inference mechanisms or libraries.
- Allow modules to be modified or replaced at run time.
- Support multiple programming paradigms.
- Provide run-time type-checking instead of compile-time checking based on syntactic types.
- Include editors and other development tools as part of the run-time system.

Some of the specialised languages developed to support AI research and applications include Prolog, Scheme, Smalltalk, OPS-5 and other production system interpreters, several varieties of Lisp including Common Lisp, Pop2 and its derivatives, e.g. Pop-11, hybrid systems supporting more than one language, e.g. Loglisp, Poplog.

Using these and other languages, various tools have been developed to support knowledge acquisition and testing, theorem provers, planners, problem solvers, parsers and other forms of software for manipulating natural language, neural net toolkits, image processing tools, robot development tools, tools for designing and testing cognitively rich agents, tools for developing multi-agent systems, rule induction and learning systems, automatic program generating and testing tools, tools for doing experiments in artificial life, and tools for supporting evolutionary computation.

Some of the tools are closely related to particular theories, or intended to support particular types of techniques, e.g. constraint-manipulation toolkits, tools for building cognitive models based on SOAR, or ACT-R.

There have been some experiments in designing new forms of hardware to support AI, e.g. hardware tailored to playing chess, hardware for vision, hardware for implementing AI languages like Lisp or Prolog, hardware for neural computation, in addition to robots and robot components. In future there may be AI models or applications using entirely new forms of computers, e.g. quantum computers or DNA computers.

Many of the tools required for AI as engineering overlap with those required for AI as science, since the task of producing intelligent applied systems has much in common with the task of producing models of natural intelligent systems.

A.5.1.3 Branches of AI

There are many branches of Artificial Intelligence including:

- **Neural Networks** – These are systems that attempt to simulate intelligence by reproducing the types of physical connections that occur in animal brains.
- **Natural Language Processing** – This involves programming computers to understand natural human languages.
- **Robotics** – This field attempts to robots to act intelligently. For example to see and hear and react to other sensory stimuli.

- **Game Playing** – This involves programming computers to play games such as chess.
- **Expert Systems** – This is where computers are programmed to make decisions in real-life situations.

A.5.2 Expert Systems

A.5.2.1 Expert Systems Presentation

Definitions of expert systems vary. Some definitions are based on function. Some definitions are based on structure. Some definitions have both functional and structural components. Many early definitions assume rule-based reasoning.

A.5.2.1.1 Functional Components

What the system does (rather than how)

> "... a computer program that behaves like a human expert in some useful ways." [Winston and Prendergast, 1984, p.6]

Problem area:

- Solve problems efficiently and effectively in a narrow problem area.
- Typically, pertains to problems that can be symbolically represented.

Problem difficulty:

- Apply expert knowledge to difficult real world problems.
- Solve problems that are difficult enough to require significant human expertise for their solution.
- Address problems normally thought to require human specialists for their solution.

Performance requirement:

- The ability to perform at the level of an expert.
- Programs that mimic the advice-giving capabilities of human experts.
- Matches a competent level of human expertise in a particular field.
- Can offer intelligent advice or make an intelligent decision about a processing function.
- Allows a user to access this expertise in a way similar to that in which he might consult a human expert, with a similar result.

Explain reasoning:

- The capability of the system, on demand, to justify its own line of reasoning in a manner directly intelligible to the enquirer.
- Incorporation of explanation processes.

A.5.2.1.2 Structural Components

Use AI techniques:

- Using the programming techniques of artificial intelligence, especially those techniques developed for problem solving.

Knowledge component:

- The embodiment within a computer of a knowledge-based component, from an expert skill.
- A computer based system in which representations of expertise are stored.

The knowledge of an expert system consists of facts and heuristics. The 'facts' constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in the field.

Expert systems are sophisticated computer programs that manipulate knowledge to solve problems.

Separate knowledge and control:

- Make domain knowledge explicit and separate from the rest of the system.

Use inference procedures – heuristics – uncertainty:

- An intelligent computer program that uses knowledge and inference procedures.

The style adopted to attain these characteristics is rule-based programming.

Exhibit intelligent behavior by skillful application of heuristics.

The 'heuristics' are mostly private, little rules of good judgment (rules of plausible reasoning, rules of good guessing) that characterize expert-level decision making in the field:

- Incorporation of ... ways of handling uncertainty.

Model human expert can be thought of as a model of the expertise of the best practitioners in the field:

- Representation of domain-specific knowledge in the manner in which the expert thinks.
- Involving the use of appropriate information acquired previously from human experts.

A.5.2.1.3 How do People Reason?

They create categories (Cash is a Current Asset and a Current Asset is an Asset)

They use specific rules, a priori rules

e.g. tax law . . . so much for each deduction

Rules can be cascaded

"If A then B" . . .

"If B then C"

A—>B—>C

They Use Heuristics – "rules of thumb"

Heuristics can be captured using rules

"If the meal includes red meat

Then choose red wine"

Heuristics represent conventional wisdom

They use past experience – "cases"

Particularly evident in precedence-based reasoning

e.g. law or choice of accounting principles

Similarity of current case to previous cases provides basis for action choice

Store cases using key attributes

Cars may be characterized by: year of car; make of car; speed of car, etc.

What makes good argumentation also makes good reasoning

They use "Expectations"

"You are not yourself today"

If we differ from expectations then it is recognized

"Patterns of behavior"

A.5.2.1.4 How do Computers Reason?

- **Computer models are based on our models of human reasoning**
- **Frames:**
    - Frame attributes called "slots".
    - Each frame is a node in one or more "isa" hierarchies.

They use rules A—>B—>C

Auditing, tax…

Set of rules is called knowledge base or rule base

They use cases (Tax reasoning and tax cases where set of cases is called a case base)

They use pattern recognition/expectations (Credit card system and data base security system):

- A network of nodes and relations in some ways very similar to a traditional database and in other ways very different. Attributes called "slots", and value can be stated explicitly.
- A method for determining the value rather than the value itself .
- Each frame is a node in one or more "isa" hierarchies.
- Higher levels general concepts – lower levels specific.
- Unspecified value can be inherited from the more general node.
- Concept: prototypical representation with defaults that may be overridden.

- **Rule Based Reasoning**

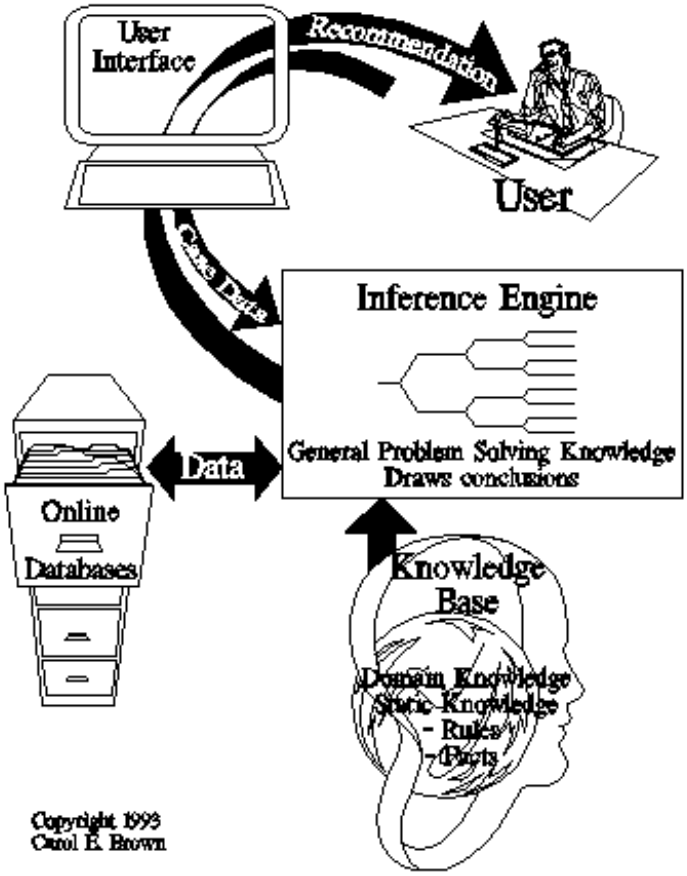

Figure A.12: Structure of a Rule-Based Expert System.

Currently, the most common form of expert system is:

- User Interface Friendly.
- Maybe "Intelligent".
- Knowledge of how to present information.
- Knowledge of user preferences...possibly accumulate with use.
- Databases.
- Contains some of the data of interest to the system.
- May be connected to on-line company or public database.
- Human user may be considered a database.

Inference Engine:

- General problem-solving knowledge or methods.
- Interpreter analyzes and processes the rules.
- Scheduler determines which rule to look at next.
- The search portion of a rule-based system.

- Takes advantage of heuristic information.
- Otherwise, the time to solve a problem could become prohibitively long.
- This problem is called the combinatorial explosion.
- Expert-system shell provides customizable inference engine.

Knowledge Base (rule base):

- Contains much of the problem solving knowledge.

Rules are of the form IF condition THEN action:

- Condition portion of the rule is usually a fact – (If some particular fact is in the database then perform this action).
- Action portion of the rule can include.
- Actions that affect the outside world (print a message on the terminal).
- Test another rule (check rule no. 58 next).
- Add a new fact to the database (If it is raining then roads are wet).

Rules can be specific, a priori rules (e.g. tax law…so much for each exemption) – represent laws and codified rules.

Rules can be heuristics (e.g. If the meal includes red meat then choose red wine). "rules of thumb" – represent conventional wisdom.

Rules can be chained together (e.g. "If A then B" "If B then C" since A—>B— >C so "If A then C") (If it is raining then roads are wet. If roads are wet then roads are slick.).

Certainty factors represent the confidence one has that a fact is true or a rule is valid.

- **Knowledge Engineering**

The discipline of building expert systems.

Knowledge acquisition:

- The process of acquiring the knowledge from human experts or other sources.
- Can involve developing knowledge to solve the problem.
- Knowledge elicitation.
- Coaxing information out of human experts.

Knowledge representation:

- Method used to encode the knowledge for use by the expert system.
- Common knowledge representation methods include rules, frames, and cases.
- Putting the knowledge into rules or cases or patterns is the knowledge representation process.

- **Case-Based Reasoning**

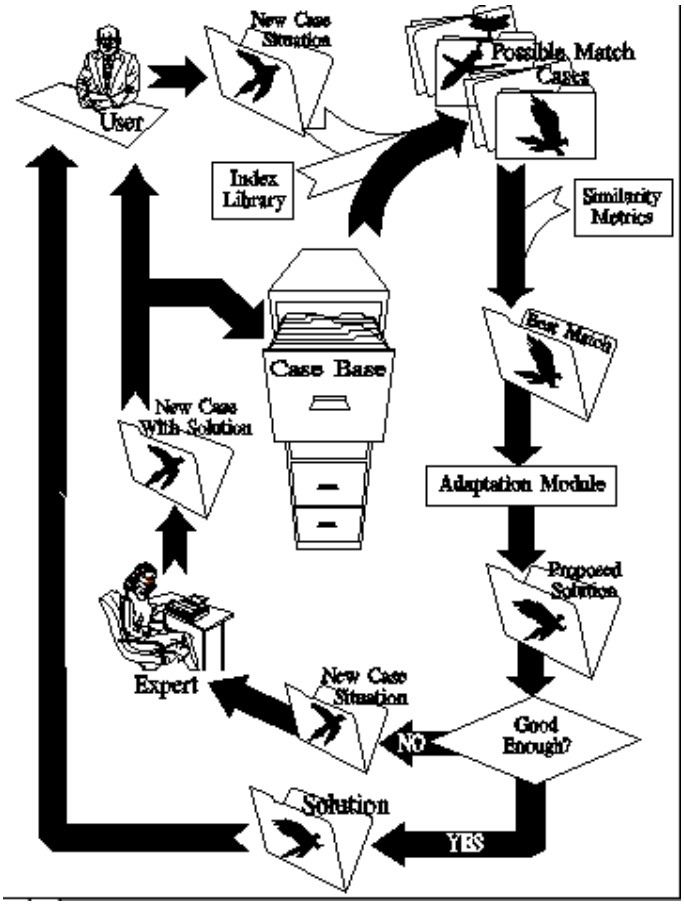

Figure A.13: The Case-Based Reasoning Process.

Uses past experiences.

Based on the premise that human beings use analogical reasoning or experiential reasoning to learn and solve complex problems.

Particularly evident in precedence-based reasoning (e.g. tax law or choice of accounting principles).

Useful when little evidence is available or information is incomplete.

Cases consist of:

- Information about the situation.
- The solution.
- The results of using that solution.
- Key attributes that can be used for quickly searching for similar patterns of attributes.

Elements in a case-based reasoning system:

- The case base – set of cases.
- The index library – used to efficiently search and quickly retrieve cases that are most appropriate or similar to the current problem.

- Similarity metrics – used to measure how similar the current problem is to the past cases selected by searching the index library.
- The adaption module – creates a solution for the current problem by either modifying the solution (structural adaptation) or creating a new solution using the same process as was used in the similar past case (derivational adaptation).

Learning:

- If no reasonably appropriate prior case is found then the current case and its human created solution can be added to the case base thus allowing the system to learn.

A.5.2.2 Advantages and Disadvantages

A.5.2.2.1 Advantages of Expert Systems

Permanence – Expert systems do not forget, but human experts may.

Reproducibility – Many copies of an expert system can be made, but training new human experts is time-consuming and expensive.

If there is a maze of rules (e.g. tax and auditing), then the expert system can "unravel" the maze.

Efficiency – can increase throughput and decrease personnel costs.

Although expert systems are expensive to build and maintain, they are inexpensive to operate.

Development and maintenance costs can be spread over many users.

The overall cost can be quite reasonable when compared to expensive and scarce human experts.

Consistency – With expert systems similar transactions handled in the same way. The system will make comparable recommendations for like situations.

Humans are influenced by:

- Recency effects (most recent information having a disproportionate impact on judgment).
- Primacy effects (early information dominates the judgment).

Documentation – An expert system can provide permanent documentation of the decision process.

Completeness – An expert system can review all the transactions, a human expert can only review a sample.

Timeliness – Fraud and/or errors can be prevented. Information is available sooner for decision making.

Breadth – The knowledge of multiple human experts can be combined to give a system more breadth that a single person is likely to achieve.

Reduce risk of doing business.

Consistency of decision making.

Documentation.

Achieve Expertise.

Entry barriers – Expert systems can help a firm create entry barriers for potential competitors.

Differentiation – In some cases, an expert system can differentiate a product or can be related to the focus of the firm.

Computer programs are best in those situations where there is a structure that is noted as previously existing or can be elicited.

A.5.2.2.2 Disadvantages of Rule-Based Expert Systems

Common sense – In addition to a great deal of technical knowledge, human experts have common sense**.** It is not yet known how to give expert systems common sense.

Creativity – Human experts can respond creatively to unusual situations, expert systems cannot.

Learning – Human experts automatically adapt to changing environments; expert systems must be explicitly updated. Case-based reasoning and neural networks are methods that can incorporate learning.

Sensory Experience – Human experts have available to them a wide range of sensory experience; expert systems are currently dependent on symbolic input.

Degradation – Expert systems are not good at recognizing when no answer exists or when the problem is outside their area of expertise.

A.5.2.3 Expert Systems Products

A.5.2.3.1 ILOG Business Rules

ILOG is the technology leader in high-performance rule engines that automate decision-making within many e-business applications. In today's constantly changing, short-turnaround Web environments, using ILOG components ensures that applications can adapt quickly, evolving ahead of the competition.

Business and e-business applications require tremendous flexibility in order to adapt to customer demands, regulatory changes and competition.

ILOG provides this flexibility with the most robust, versatile business rule management software available for C++ and Java.

Build applications that last with ILOG JRules and ILOG RulesThe Java and C++ rule engines of ILOG JRules and ILOG Rules share a common set of tools known as the Rule Kit. The Rule Kit includes:

- Rule Repository Open, extensible business rule repository for centrally storing and managing business rules, and deploying them throughout the enterprise.
- Rule Builder Integrated development environment for developing and debugging business rule applications.
- Business Rule Language Customizable and extensible business rule language, placing business rule power in the hands of business users.
- Rule Editor Powerful, adaptable Web-enabled and JavaBean components that can be embedded in applications.

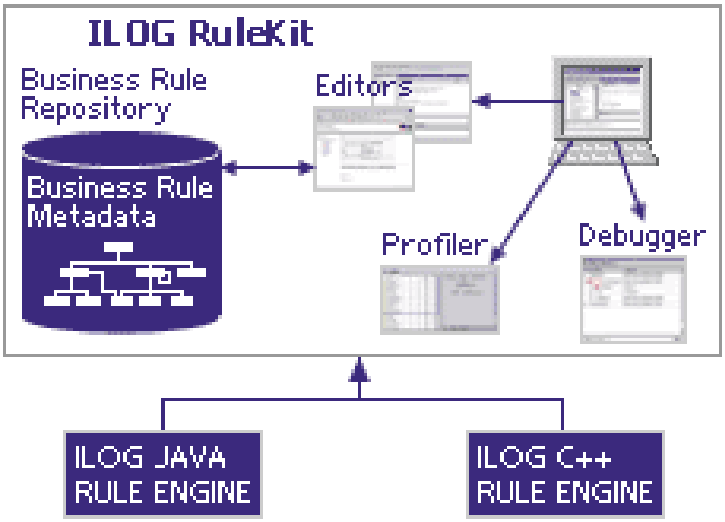

Figure A.14: ILOG Architecture.

Vendor: http://www.ilog.com

A.5.3 Neural Networks

A.5.3.1 Brief History of Neural Networks

The earliest work in neural computing goes back to the 1940s when McCulloch and Pitts introduced the first neural network computing model. In the 1950s, Rosenblatt's work resulted in a two-layer network, the perceptron, which was capable of learning certain classifications by adjusting connection weights. Although the perceptron was successful in classifying certain patterns, it had a number of limitations. The perceptron was not able to solve the classic XOR (exclusive or) problem. Such limitations led to the decline of the field of neural networks. However, the perceptron had laid foundations for later work in neural computing.

In the early 1980s, researchers showed renewed interest in neural networks. Recent work includes Boltzmann machines, Hopfield nets, competitive learning models, multilayer networks, and adaptive resonance theory models.

A.5.3.2 Artificial Neural Networks and Connectionist Models

Based on pattern recognition – used for credit assessment and fraud detection.

A set of interconnected relatively simple mathematical processing elements.

Looks for patterns in a set of examples and learns from those examples by adjusting the weights of the connections to produce output patterns.

Input to output pattern associations are used to classify a new set of examples.

Able to recognize patterns even when the data is noisy, ambiguous, distorted, or has a lot of variation.

Neural network construction and training:

- The architecture used (e.g. feed-forward).

- How the neurons are organized (e.g. an input layer with five neurons, two hidden layers with three neurons each, and an output layer with two neurons).
- The state function used (e.g. summation function).
- The transfer functions used (e.g. sigmoid squashing function).
- The training algorithm used (e.g. back-propagation).

Architecture shows how the processing elements are connected.

Commonly used architectures:

- Feed-forward.

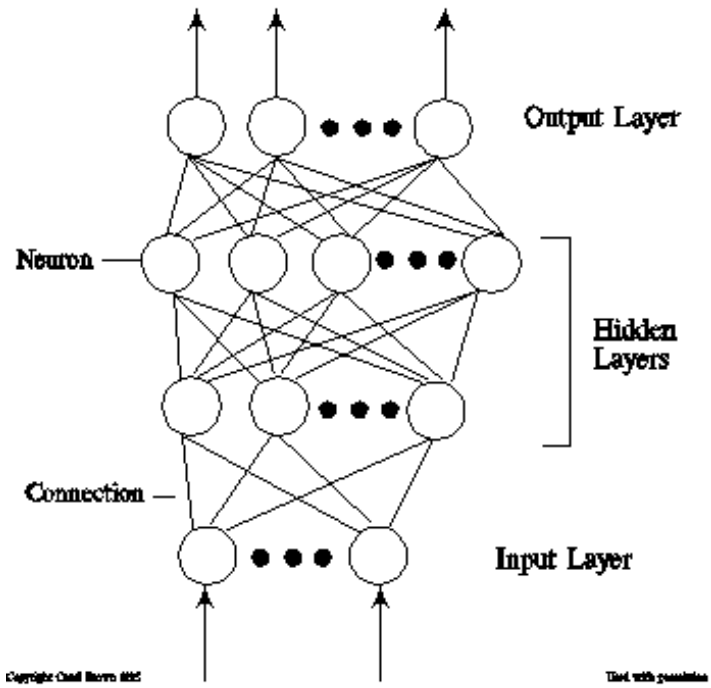

Figure A.15: Feed-Forward Neural Network Structure.

Feed-Forward Neural Network Structure is organized into a series of layers.

Layers (also called levels, fields or slabs):

- Input layer.
- One or more hidden layers.
- Output layer.

Some consider the number of layers to be part of architecture. Others consider the number of layers and nodes per layer to be attributes of the network rather than part of the architecture.

Neurons – the processing elements.

The vocabulary in this area is not completely consistent and different authors tend to use one of a small set of terms for a particular concept.

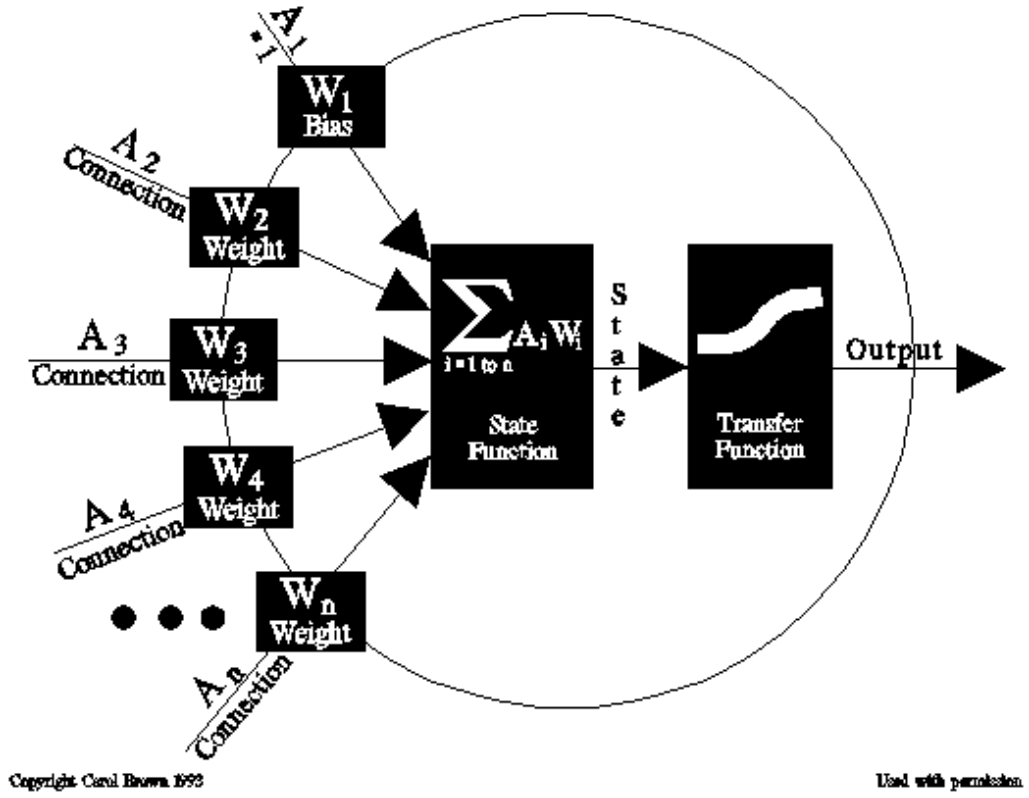

Figure A.16: Structure of a Neuron.

Structure of a Neuron consists of:

- A set of weighted input connections;
- A bias input;
- A state function;
- A non-linear transfer function; and
- An output.

Input connections have an input value that is either received from the previous neuron or in the case of the input layer from the outside.

Bias is not connected to the other neurons in the network and is assumed to have an input value of 1 for the summation function.

Weights

- A real number representing the strength or importance of an input connection to a neuron.
- Each neuron input, including the bias, has an associated weight.

State function

- The most common form is a simple summation function.
- The output of the state function becomes the input for the transfer function.

Transfer function

- A non-linear mathematical function used to convert data to a specific scale.
- Two basic types of transfer functions: continuous and discrete.
- Commonly used continuous functions used are Ramp, Sigmoid, Arc Tangent and Hyperbolic Tangent.
- Continuous functions sometimes called squashing functions.
- Commonly used discrete functions are Step and Threshold.
- Discrete transfer function sometimes called activation function.

Training

- The process of using examples to develop a neural network that associates the input pattern with the correct answer.
- A set of examples (training set) with known outputs (targets) is repeatedly fed into the network to "train" the network.
- This training process continues until the difference between the input and output patterns for the training set reaches an acceptable value.
- Several algorithms used for training networks.
- Most common is back-propagation.
- Back-propagation is done is two passes.
- First the inputs are sent forward through the network to produce an output.
- Then the difference between the actual and desired outputs produces error signals that are sent "backwards" through the network to modify the weights of the inputs.

A.5.3.3 Neural Networks Products

A.5.3.3.1 NeuroLab for Extend

Key features: NeuroLab is a neural network library for Extend (a graphical icon-based simulation program) and is a powerful and effective tool for understanding, designing and simulating artificial neural network systems. NeuroLab consists of more than 70 functional blocks and many example models. Implemented neural networks are Hopfield (regular and sparsely), perceptron, competitive (supervised and unsupervised), recurrent, context, feature map (2D to 1D and 2D to 2D), Boltzmann machine and single/ multi-layer feed-forward networks.

Each functional block in NeuroLab is easy to use and understand because of its intuitive appearance and interactive dialog box. Users can modify the network's parameters during simulation through each functional block's dialog box. NeuroLab is applicable in any field where non-linear mapping and adaptive capability are integral parts of problem solving. Major applicable fields are image processing, data classification, future prediction, adaptive control, dynamics identification, optimization and content addressable memory. NeuroLab can be used for Control of an Inverted Pendulum, Chinese Character Restoration, Exclusive OR, and Stock Prediction.

OS: Windows, Macintosh

Vendor: NeuroLab Department. http://www.mikuni.com

A.5.3.3.2 STATISTICA Neural Networks

STATISTICA Neural Networks is a comprehensive, state-of-the-art, powerful, and extremely fast neural network data analysis package, featuring:

Integrated pre- and post-processing, including data selection, nominal-value encoding, scaling, normalization and missing value substitution, with interpretation for classification, regression and time series problems. Exceptional ease of use coupled with unsurpassed analytic power; for example, a unique wizard-style Intelligent Problem Solver can walk the user step-by-step through the procedure of creating a variety of different networks and choosing the network with the best performance (a task that would otherwise require a lengthy "trial and error" process and a solid background in the underlying theory).

Powerful exploratory and analytic techniques, including Input Feature Selection algorithms (choosing the right input variables in exploratory data analysis -which is a typical application of neural networks is often a time-consuming process; STATISTICA Neural Networks can also do this for the user).

State-of-the-art, highly optimized training algorithms (including Conjugate Gradient Descent and Levenberg-Marquardt); full control over all aspects that influence the network performance such as activation and error functions, or network complexity. Support for combinations of networks and network architectures of practically unlimited sizes organized in Network Sets; selective training of network segments; merging, saving of network sets in separate files. API support for embedded solutions using Visual Basic, Delphi, C, C++ and other languages.

OS: Windows 95, Windows NT

Vendor: StatSoft. http://www.statsoft.com/stat nn.html

A.5.3.3.3 BIOSIM (Non-Commercial Software)

BIOSIM is a biologically-oriented neural network simulator. It implements four neuron models: a simple model only switching ion channels on and off, the original Hodgkin-Huxley model, the SWIM model (a modified HH model) and the Golowasch-Buchholz model (the most enhanced model). Dendrites consist of a chain of segments without bifurcation. BIOSIM includes a graphical user interface and was designed for research and teaching.

OS: information not available

Contact: Stefan Bergdoll Department of Software Engineering (ZXA/US), BASF Inc. 67056 Ludwigshafen, Germany Phone: +49-621-60-21372

Download: ftp://ftp.uni-kl.de/pub/bio/neurobio/

A.5.4 Fuzzy Logic

A.5.4.1 Fuzzy Logic Presentation

Fuzzy logic is an area of research based on the work of L.A. Zadeh. It is a departure from classical two-valued sets and logic, that uses "soft" linguistic (e.g. large, hot, tall) system variables and a continuous range of truth values in the interval [0,1], rather than strict binary (True or False) decisions and assignments. Fuzzy logic is used where a system is difficult to model exactly (but an inexact model is available), is controlled by a human operator or expert, or where ambiguity or vagueness is common. A typical fuzzy system consists of a rule base, membership functions, and an inference procedure.

A.5.4.2 Fuzzy Logic Products

A.5.4.2.1 DataEngine

DataEngine is the software for intelligent data analysis and data mining. By using neural networks, fuzzy logic and statistical methods, DataEngine provides you with the most advanced techniques for data analysis. Its flexible structure, interfaces, and powerful visualization module make it a must for everybody involved with complex data analysis tasks.

DataEngine can be extended by the user via user-defined function blocks. A DLL-based PlugIn interface enables you to integrate your own analysis tools into DataEngine in a simple and effective way. A number of examples of PlugIns are provided with this software.

OS: Windows 95, Windows 98, Windows NT

Vendor: MIT – Management Intelligenter Technologien GmbH http://www.mitgmbh.de

Demo: http://www.mitgmbh.de/mit/sp/demos/index.htm

A.5.4.2.2 Rule Maker

Key features: Rule Maker is an automatic rule generator add-on for CubiCalc or CubiCalc RTC. It uses neural networks and statistics to create variables, fuzzy sets, and rules automatically from data you supply. The product offers a choice of several methods. If you want to use fuzzy sets that you already have, Rule Maker can build a system from them. If you want build a system from scratch, it can create regularly-spaced fuzzy sets or place them according to clusters in your data. Either way, the resulting system is a full-fledged CubiCalc project.

Unlike purely neural-based automatic generation facilities, Rule Maker also handles sparse data sets. It can build a reasonable system with just a handful of data points. Rule Maker also includes new techniques invented at HyperLogic not available anywhere else.

OS: Windows 3.1 or higher, requires CubiCalc or CubiCalc RTC

Vendor: HyperLogic Corporation http://www.hyperlogic.com

A.5.4.2.3 WINROSA

Key features: WINROSA is a software that complements many tools already available for building fuzzy logic systems. Setting up a fuzzy rule base can be a very difficult task if there is no expert knowledge available regarding the process your modeling. WINROSA can assist you by automatically generating fuzzy rules from your data. It is based on the fuzzy ROSA method (Rule Oriented Statistical Analysis), which has been developed at the University of Dortmund. If you want to find relationships in your data and represent them as fuzzy rules, or if you wish to verify rule bases, then WINROSA is a must. WINROSA outstrips the performance of existing fuzzy sets completely.

OS: Windows 95, Windows NT

Vendor: MIT – Management Intelligenter Technologien GmbH http://www.mitgmbh.de

A.5.4.3 Genetic Algorithms

A.5.4.3.1 Genetic Algorithms Presentation

The Genetic Algorithm is a model of machine learning which derives its behavior from a metaphor of some of the mechanisms of evolution in nature. This is done by the creation within a machine of a population of individuals represented by chromosomes, in essence a set of character strings that are analogous to the base-4 chromosomes that we see in our own DNA. The individuals in the population then go through a process of simulated "evolution". Genetic algorithms are used for a number of different application areas. An example of this would be multidimensional optimization problems in which the character string of the chromosome can be used to encode the values for the different parameters being optimized. In practice, therefore, we can implement this genetic model of computation by having arrays of bits or characters to represent the chromosomes. Simple bit manipulation operations allow the implementation of crossover, mutation and other operations. Although a substantial amount of research has been performed on variable- length strings and other structures, the majority of work with genetic algorithms is focussed on fixed-length character strings. We should focus on both this aspect of fixed-length and the need to encode the representation of the solution being sought as a character string, since these are crucial aspects that distinguish genetic programming, which does not have a fixed length representation and there is typically no encoding of the problem. When the genetic algorithm is implemented it is usually done in a manner that involves the following cycle: Evaluate the fitness of all of the individuals in the population. Create a new population by performing operations such as crossover, fitness-proportionate reproduction and mutation on the individuals whose fitness has just been measured. Discard the old population and iterate using the new population. One iteration of this loop is referred to as a generation. There is no theoretical reason for this as an implementation model. Indeed, we do not see this punctuated behavior in populations in nature as a whole, but it is a convenient implementation model. The first generation (generation 0) of this process operates on a population of randomly generated individuals. From there on, the genetic operations, in concert with the fitness measure, operate to improve the population.

A.5.4.4 Genetic Algorithms Products

A.5.4.4.1 Evolver

Evolver is an optimization add-in for Microsoft Excel. Evolver uses innovative genetic algorithm (GA) technology to quickly solve complex optimization problems in finance, distribution, scheduling, resource allocation, manufacturing, budgeting, engineering, and more. Virtually any type of problem that can be modeled in Excel can be solved by Evolver, including previously unsolvable problems. Evolver requires no knowledge of programming or GA theory and ships with a fully illustrated manual, several examples, and free, unlimited technical support. Evolver is available in three versions: Standard, Professional, and Industrial. The Professional and Industrial versions have increased problem capacities and advanced features, including the Evolver Developers Kit.

OS: Windows 3.x, Windows 95, Windows 98, Windows NT

Microsoft Excel 5.0 or higher
Vendor: Palisade Corporation. http://www.palisade.com

Demo: http://www.palisade.com/html/trial—versions.html

A.5.4.4.2 GECO (Non-Commercial Software)

GECO (Genetic Evolution through Combination of Objects) is an extensible, object-oriented framework for prototyping GENETIC ALGORITHMs in Common Lisp. GECO makes extensive use of CLOS,

the Common Lisp Object System, to implement its functionality. The abstractions provided by the classes have been chosen with the intent both of being easily understandable to anyone familiar with the paradigm of genetic algorithms, and of providing the algorithm developer with the ability to customize all aspects of its operation. It comes with extensive documentation, in the form of a PostScript file, and some simple examples are also provided to illustrate its intended use.

OS: information not available. Contact: George P.W. Williams, Jr. 1334 Columbus City Rd. Scottsboro, AL 35768, U.S.A. e-mail: george@hsvaic.hv.boeing.com

Download: ftp://ftp.aic.nrl.navy.mil/pub/galist/src/

A.6 REFERENCES

A.6.1 Artificial Intelligence

Expert Systems in Finance, Daniel E. O'Leary and Paul R. Watkins (Editors), 1992.

Expert Systems in Business and Finance: Issues and Applications, Paul R. Watkins and Lance B. Elliot (Editors), 1993.

Artificial Intelligence in Accounting and Auditing: Using Expert Systems, Volume II, Miklos A. Vasarhelyi and B.N. Srinidhi (Editors), 1993.

Expert Systems introduction, Zbigniew Swiatnicki, 1998.

Neural Networks introduction, Zbigniew Swiatnicki and Roman Wantoch-Rekowski, 1999.

A.6.2 Simulation

Techniques for Monte Carlo Optimizing, Arsham, H., Monte Carlo Methods and Applications, 4(3), 181-230, 1998.

Stochastic Optimization of Discrete Event Systems Simulation, Arsham, H., Microelectronics and Reliability, 36(10), 1357-1368, 1996.

Conditional Monte Carlo: Gradient Estimation and Optimization Applications, Fu, M. and Hu, J-Q., Kluwer Academic Publishers, 1997.

Estimating the optimum of a stochastic system using simulation, Rollans, S. and McLeish, D., Journal of Statistical Computation and Simulation, 72, 357-377, 2002.

Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method, Rubinstein, R. and Shapiro, A., John Wiley & Sons, 1993.

Bossel, H., *Modeling & Simulation*, Peters, A.K., Pub., 1994. Delaney W., and Vaccari, E., *Dynamic Models and Discrete Event Simulation*, Dekker, 1989.

Fishman, G., Discrete-Event Simulation: Modeling, Programming and Analysis, Springer-Verlag, Berlin, 2001.

Fishwick, P., Simulation Model Design and Execution: Building Digital Worlds, Prentice-Hall, Englewood Cliffs, 1995.

Ghosh, S. and Lee, T., Modeling & Asynchronous Distributed Simulation: Analyzing Complex Systems, IEEE Publications, 2000.

Gimblett, R., Integrating Geographic Information Systems and Agent-Based Modeling: Techniques for Simulating Social and Ecological Processes, Oxford University Press, 2002.

Harrington, J. and Tumay, K., Simulation Modeling Methods: An Interactive Guide to Results-Based Decision, McGraw-Hill, 1998.

Haas, P., Stochastic Petri Net Models Modeling and Simulation, Springer Verlag, 2002.

Hill, D., Object-Oriented Analysis and Simulation Modeling, Addison-Wesley, 1996.

Kouikoglou, V. and Phillis, Y., *Hybrid Simulation Models of Production Networks*, Kluwer Pub., 2001.

Law, A. and Kelton, W., *Simulation Modeling and Analysis*, McGraw-Hill, 2000.

Nelson, B., Stochastic Modeling: Analysis & Simulation, McGraw-Hill, 1995.

Oakshott, L., *Business Modelling and Simulation*, Pitman Publishing, London, 1997.

Pidd, M., Computer Simulation in Management Science, Wiley, 1998.

Rubinstein, R. and Melamed, B., *Modern Simulation and Modeling*, Wiley, 1998.

Severance, F., System Modeling and Simulation: An Introduction, Wiley, 2001.

Van den Bosch, P. and Van der Klauw, A., *Modeling, Identification & Simulation of Dynamical Systems*, CRC Press, 1994.

Woods, R. and Lawrence, K., *Modeling and Simulation of Dynamic Systems*, Prentice Hall, 1997.